UNIVERSIDADE FEDERAL DE SANTA CATARINA
CAMPUS ARARANGUÁ

João Paulo Cardoso de Lima

DESENVOLVIMENTO DE SERVIDORES PARA LABORATÓRIOS REMOTOS BASEADO NO PARADIGMA DE DISPOSITIVOS INTELIGENTES

Araranguá

2016

João Paulo Cardoso de Lima

DESENVOLVIMENTO DE SERVIDORES PARA LABORATÓRIOS REMOTOS BASEADO NO PARADIGMA DE DISPOSITIVOS INTELIGENTES

Trabalho de Conclusão de Curso submetido à Universidade Federal de Santa Catarina, como parte dos requisitos necessários para a obtenção do Grau de Bacharel em Engenharia de Computação.

Araranguá, dezembro de 2016.

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Lima, João Paulo Cardoso de
Desenvolvimento de servidores para laboratórios remotos baseado no paradigma de dispositivos inteligentes / João Paulo Cardoso de Lima ; orientador, Juarez Bento da Silva - Araranguá, SC, 2016.
87 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Campus Araranguá. Graduação em Engenharia de Computação.

Inclui referências

1. Engenharia de Computação. 2. Laboratórios remotos. 3. Arquitetura orientada a serviço. 4. Reuso de software. I. Bento da Silva, Juarez . II. Universidade Federal de Santa Catarina. Graduação em Engenharia de Computação. III. Título.

João Paulo Cardoso de Lima

DESENVOLVIMENTO DE SERVIDORES PARA LABORATÓRIOS REMOTOS BASEADO NO PARADIGMA DE DISPOSITIVOS INTELIGENTES

Este Trabalho de Conclusão de Curso foi julgado aprovado para a obtenção do Título de "Bacharel em Engenharia de Computação", e aprovado em sua forma final pela Curso de Graduação em Engenharia de Computação.

Araranguá, 09 de dezembro 2016.

Prof. Dr. Anderson Fernandes Perez
Coordenador do Curso

Banca Examinadora:

Prof. Dr. Juarez Bento da Silva
Orientador

Prof. Dr. João Bosco da Mota Alves

Dr. Paulo Manoel Mafra

À minha família, amigos e professores, pelo incentivo, apoio e compreensão durante toda minha jornada.

AGRADECIMENTOS

Agradeço a todos que contribuíram direta ou indiretamente com este trabalho, em especial aos meus pais, irmãos e amigos pelo suporte e incentivo em todos momentos que precisei. Ao professor Juarez Bento da Silva, pela orientação e incentivo durante os últimos 4 anos de minha graduação. A todos professores que contribuíram para minha formação, que além de ensinar o conteúdo de sua formação também me incentivaram e mostraram caminhos. Ao Laboratório de Experimentação Remota e sua equipe, pelos desafios, pelos trabalhos em grupo e pelas amizades. À Universidade Federal de Santa Catarina e aos seus professores e servidores técnico-administrativo, pela infraestrutura, pelo ensino e pela assistência. Ao Conselho Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (Capes), pelo suporte financeiro em diferentes projetos que pude participar. Aos contribuintes brasileiros, por garantirem a oportunidade de estudar em instituição pública durante toda minha vida.

Nenhum homem pisa no mesmo rio duas vezes, porque não é o mesmo rio e ele não é o mesmo homem.

Heráclito de Éfeso

RESUMO

Os laboratórios remotos são reconhecidos por estender o acesso a laboratórios reais. A possibilidade de compartilhá-los entre diversas instituições têm despertado o interesse da comunidade acadêmica na busca de uma infraestrutura interoperável e flexível. No entanto, a falta de padronização na área é um fator limitante para estratégias que reduzam o tempo de desenvolvimento e promovam a integração com outros sistemas. Neste trabalho é proposta uma aplicação modelo para o desenvolvimento de laboratórios remotos e o compartilhamento em ambientes de aprendizagem baseada no paradigma de dispositivos inteligentes. Para sua execução foram identificados modelos de referência e, então, projetadas partes de um protótipo com base no modelo de Laboratório como um Serviço e no paradigma de dispositivos inteligentes. O trabalho ocorre dentro do contexto do Grupo de Trabalho em Experimentação Remota Móvel (GT-MRE), portanto são apresentadas arquitetura de software e infraestrutura da plataforma, além de serviços para controle de concorrência de usuários. A avaliação da aplicação foi realizada através de medidas de tempo de resposta para execução de serviços no sistema de escalonamento e de recuperação de dados de sensor na aplicação de servidor. A aplicação mostrou-se adequada para implantação em *Single Board Computers* e foram criadas aplicações clientes com sucesso. Assim, este trabalho colaborou com a padronização e reuso de software em vários níveis para os laboratórios do GT-MRE e seu sistema de acesso.

Palavras-chave: sistemas de gerenciamento de laboratórios remotos, reuso de software, paradigma de dispositivos inteligentes, servidor de laboratório

ABSTRACT

Remote laboratories are known for extending access to real laboratories. The possibility of sharing them among several institutions has motivated the academic community in the search for interoperable and flexible infrastructures. However, the lack of standardization in the area is a limiting factor for strategies capable of reducing development time and promoting integration with other systems. In this work, a template application for development of remote labs and sharing in learning environments based on the smart device paradigm is proposed. In order to achieve this goal, reference models were identified and parts of a prototype based on the Laboratory as a Service model and on the smart device paradigm were designed. As this work is part of Working Group on Mobile Remote Experimentation's (GT-MRE) initiatives, software architecture and infrastructure of the platform are presented, as well as services to handle user concurrency. The evaluation was based on response time measurements to perform services on the scheduling system and to retrieve sensor data from the lab server. The application proved to be suitable for deployment on Single Board Computers and client applications were successfully created. Thus, this work have benefited the standardization and different levels of software reuse on lab server application and platform provided by GT-MRE.

Keywords: remote lab management system, software reuse, smart device paradigm, lab server application.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

SUMÁRIO

1 INTRODUÇÃO

Os laboratórios remotos são amplamente conhecidos por estender o acesso a laboratórios reais por meio de tecnologias de comunicação e controle. Este tipo de laboratório é uma tendência no ensino de engenharia e ciências, e tem como objetivo permitir o uso de equipamentos de laboratórios de forma organizada e remota (COOPER; FERREIRA, 2009).

O uso destes laboratórios possibilita oferecer práticas laboratoriais para estudantes sem condições de frequentar um laboratório convencional, sejam estas condições causadas pela falta de recursos na sua instituição, por tempo disponível ou por limitações físicas do próprio usuário. Além disso, essa técnica também permite reduzir os custos para condução de pesquisas através do compartilhamento de recursos caros entre diversos pesquisadores (SANTANA et al., 2013).

O início desse tipo de acesso foi introduzido entre os anos de 1993 e 1995 como forma de ampliar o acesso a equipamentos laboratoriais caros em disciplinas de engenharia elétrica (ALVES et al., 2007). Os primeiros relatos que aparecem na literatura exploraram o potencial da sua abordagem para a educação a distância, em especial por fornecer atividades de laboratório para estudantes de localidades distantes (AKTAN et al., 1996; BOHUS et al., 1995).

Desde então, diversos projetos surgiram a fim de desenvolver e popularizar uma infraestrutura adequada para suporte aos laboratórios. Alguns projetos proeminentes que podem ser destacados são: iLab (HARWARD et al., 2008), LabShare (LOWE; MACHET; KOSTULSKI, 2012), VISIR[1] (GUSTAVSSON et al., 2007) e WebLabDeusto[2] (ORDUÑA et al., 2011). Ao mesmo tempo, outros trabalhos contribuíram com o desenvolvimento de repositórios que indexam recursos de diferentes instituições, como o Lab2Go (ZUTIN et al., 2010) e LiLa (RICHTER; TETOUR; BOEHRINGER, 2011).

Um elemento chave no desenvolvimento destes projetos é um framework comum para acesso e administração dos laboratórios, comumente chamado de Sistema de Gerenciamento de Laboratórios Remotos, ou RLMS (LOWE et al., 2016). Com a crescente popularização desta modalidade de laboratórios, cada vez mais os RLMSs precisam oferecer funcionalidades para um grande conjunto de sistemas de laboratórios remotos heterogêneos que podem estar distribuídos em diferentes instituições e espalhados geograficamente. As ferramentas de gerenciamento e serviços comuns (autenticação, escalonamento e rastreamento de usuário), além de serem independentes de

[1] `http://openlabs.bth.se/electronics/index.php/en`

[2] `www.weblab.deusto.es`

um laboratório em particular, permitem que todos os laboratórios desenvolvidos com base nestes componentes possam receber atualização automaticamente.

De maneira geral, os atuais sistemas de acesso a laboratórios remotos seguem o modelo cliente-servidor em que tecnologias da Internet e da Web 2.0 são o centro do desenvolvimento dos atuais frameworks. Neste modelo são identificados três atores: (a) o desenvolvedor do sistema de acesso; (b) o criador de experimentos; e (c) o usuário. Inicialmente, esses laboratórios eram adaptações de laboratórios já existentes feitas por pesquisadores e desenvolvedores do sistema de acesso, e a este fato se deve a predominância de laboratórios para engenharia elétrica na literatura em todo o mundo (GRAVIER et al., 2008).

Cada vez mais destaca-se o papel de criadores de experimentos como indivíduos de diferentes áreas do conhecimento preocupados apenas em disponibilizar equipamentos de sua área para serem utilizados remotamente, não somente no ensino superior, mas também para o ensino básico (MAITI; MAXWELL; KIST, 2013; MAITI; KIST; MAXWELL, 2015). No entanto, para permitir que usuários criem novos aparatos e sejam provedores de laboratórios remotos são necessários sistemas de acesso mais flexíveis. Mas, antes mesmo da etapa de disponibilização, a preocupação reside na forma em que será projetado e quais ferramentas e tecnologias serão utilizadas para o desenvolvimento do servidor de laboratório.

1.1 CONTEXTUALIZAÇÃO

Uma preocupação constante no projeto de sistemas computacionais, independente do seu domínio, é a capacidade de expansão, interoperabilidade, adequação das tecnologias e atualização com o estado da arte, apenas para mencionar algumas. Já em recentes sistemas de *Internet of Things* (IoT), devido à diversidade de tecnologias utilizadas para desenvolvimento de seus dispositivos, as questões de interoperabilidade são ainda mais evidentes. Os laboratórios remotos enfrentam dificuldades semelhantes às de sistemas de IoT por dependerem de dispositivos e protocolos heterogêneos e interação com recursos reais.

Por terem sido construídos sob diferentes tecnologias, a integração de laboratórios com outros sistemas necessita de camadas de software adicionais para prover novas funcionalidades ou adequar-se a um novo protocolo. Além disso, as soluções cliente-servidor fortemente acopladas são predominantes no projeto e implementação de interação e gerenciamento de um laboratório remoto. O forte acoplamento e a falta de padronização têm dificultado a capa-

cidade de compartilhar laboratórios em diferentes plataformas de aprendizagem, bem como de operá-los colaborativamente na plataforma escolhida pelo usuário (TAWFIK et al., 2014).

Os aspectos técnicos de implementação destes sistemas para uso em larga escala e compartilhamento de recursos começaram a ser explorados nos últimos anos (JONG; SOTIRIOU; GILLET, 2014), sendo a federação de laboratórios um dos principais problemas discutidos na literatura. Os trabalhos de Orduña et al. (2014) e Lowe et al. (2016), por exemplo, descrevem como foram criados protocolos de federação de laboratórios compartilháveis entre os RLMSs WebLabDeusto, LabShare e iLabs Shared Architecture.

Já os trabalhos de Tawfik et al. (2014), Sancristobal et al. (2013) focam na abordagem de acesso aos laboratórios remotos como serviços web e componentes modulares, a fim de facilitar a manutenção e a reusabilidade dos componentes. Em uma abordagem mais profunda, Salzmann e Gillet (2013), Richter, Grube e Zutin (2012) reúnem tecnologias de outros domínios para descrição de serviço e gerenciamento inteligente dos recursos para implementar serviços de descoberta em ambientes de aprendizagem com laboratórios interoperáveis e tecnologia transparente ao usuário, por exemplo.

Na maioria dos casos, as contribuições destes autores alteram o modelo de serviço, mas a arquitetura e funcionalidades básicas permanecem as mesmas. Enquanto isso, o crescente interesse em laboratórios remotos exige estratégias de projeto de software como reuso, abstração e automação na geração de código. A falta de padronização, no entanto, é um fator limitante para estratégias e práticas comuns que reduzam o tempo de desenvolvimento de laboratórios, bem como a integração com outros sistemas.

Um novo criador de laboratório dispenderá muito esforço para atualizar-se com o estado da arte e, provavelmente, acabará implementando um sistema que atenda às necessidades de sua instituição e dificilmente atenda aos requisitos de integração com ambientes de aprendizagem de outras instituições. Sendo assim, é necessário compreender as melhores práticas para o desenvolvimento de servidores de laboratórios remotos e de integração entre os sistemas, e aderir às especificações e padronizações discutidas em instituições e grupos específicos para esse fim.

1.2 PERGUNTA DE PESQUISA

Considerando os problemas enfrentados por desenvolvedores de sistemas de laboratório remoto em relação à padronização de seus componentes e arquitetura que permita flexibilidade e desempenho, e dada as tendências levantadas pela comunidade de desenvolvimento de software, define-se a se-

guinte pergunta de pesquisa:

"Como o paradigma de dispositivos inteligentes pode colaborar com o reuso de software e compartilhamento de laboratório remotos?"

1.3 OBJETIVOS

Esta seção apresenta os objetivos geral e específicos do desenvolvimento deste trabalho.

1.3.1 Objetivo Geral

Considerando a pergunta de pesquisa, foi formulado o objetivo geral deste trabalho:

Projetar uma aplicação modelo para o desenvolvimento de laboratórios remotos e o compartilhamento em ambientes de aprendizagem.

1.3.2 Objetivos Específicos

Os objetivos deste trabalho estão relacionados aos desafios técnicos apresentados anteriormente e incluem:

- Identificar questões relacionadas ao projeto de experimentos remotos com capacidade para serem compartilhados em diferentes LMSs.
- Detalhar os requisitos e arquitetura da plataforma de experimentação do GT-MRE.
- Projetar sistema de escalonamento compatível com requisitos do GT.
- Projetar uma aplicação de servidor de laboratório baseada no paradigma de dispositivos inteligentes.

1.4 JUSTIFICATIVA E MOTIVAÇÃO

Ao levar em consideração os papéis envolvidos em sistemas de laboratórios remotos, seus desenvolvedores devem prever como os criadores irão reutilizar uma aplicação com funcionalidades comuns para criar novos laboratórios. Idealmente, os criadores preocupam-se somente com o controle do

seu equipamento de laboratório e com a interface de usuário, que estará estritamente ligada a aspectos pedagógicos e de usabilidade do laboratório.

Para isso, o modelo de interação entre os componentes do laboratório remoto deve ser bem definido, prefencialmente seguindo protocolos recomendados pela comunidade de desenvolvedores e, se possível, aderindo a normas específicas desse domínio. A aplicação de padrões ao cenário da aprendizagem eletrônica permite o reuso de conteúdos e atividades didáticas através da Internet e provê ferramentas que podem ser estendidas a diferentes dispositivos e plataformas (RUIZ et al., 2013).

Nos últimos anos muitos autores têm investigado aspectos técnicos na implementação de laboratórios remotos para uso em larga escala, laboratórios distribuídos geograficamente, descrição de serviços e outros mecanismos para compartilhamento de laboratórios remotos para educação e pesquisa. Apesar de estas iniciativas buscarem estabelecer recomendações para laboratórios online, as ferramentas originadas ainda não suprem totalmente as necessidades dos criadores de laboratórios remotos sem profundo conhecimento em sistemas de gerenciamento.

Por outro lado, o uso de laboratórios remotos ainda é incipiente no Brasil. Embora existam trabalhos discutindo sobre implementação desses laboratórios e os benefícios do seu proveito na educação, poucos trabalhos discutem os desafios técnicos de escalabilidade e produção. Uma das instituições preocupadas com este tema é o RExLab da UFSC, onde está sendo desenvolvido este trabalho. O RExLab desenvolve sua pesquisa com base no conceito de experimentação remota desde 1997 com o objetivo de popularizar experimentação e o saber científico, e, assim, "promover a melhoria da qualidade da educação pela modernização do ensino em todos os seus níveis"[3].

Uma das iniciativas desenvolvidas no RExLab é o Grupo de Trabalho em Experimentação Remota Móvel (GT-MRE), do qual o presente trabalho faz parte. O GT-MRE tem apoio da Capes para financiamento de recursos e é gerido pela Rede Nacional de Pesquisa (RNP) através do edital GTs temáticos 2014 para educação a distância. Durante a primeira fase do grupo de trabalho (2014-2015), ocorreram as especificações e desenvolvimento da plataforma de experimentação remota com foco na construção de experimentos e das interfaces de usuários, mas com pouca atenção no projeto de software do servidor de laboratórios remotos.

Após a conclusão da primeira fase surgiram questões de escalabilidade e disseminação para serem resolvidas. Assim, alguns dos compromissos da segunda fase do GT-MRE relacionados com este trabalho são:

- Prover meios para integração de laboratórios remotos em diferentes

[3] http://rexlab.ufsc.br/about

ambientes de aprendizagem, bem como para indexação em repositórios.

- Permitir que outros criadores de laboratórios, sem um profundo conhecimento em software e hardware, possam reutilizar esta solução e disponibilizar seus equipamentos com propósito educacional.

Recentemente começaram a ser discutidos métodos para armazenamento e recuperação de objetos de aprendizagem para laboratórios remotos com foco em sistemas de aprendizagem inteligentes no grupo de trabalho do IEEE P1876[4]. Com base nos benefícios que a padronização de objetos de aprendizagem inteligentes conectados para laboratórios online podem trazer, surge a necessidade de adequar o serviço oferecido pelo GT-MRE de forma a permitir não somente os membros do RExLab a produzirem experimentos, mas a toda comunidade interessada.

1.5 METODOLOGIA

Para atingir os objetivos apresentados na seção 1.3, foi definida uma estratégia composta pelos seguintes passos:

1. Atualizar o conhecimento através da revisão de literatura, isto inclui periódicos e anais de conferências, e também através da análise de relatórios técnicos e documentação de padrões relacionados à área.

2. Descrever a arquitetura de software e infraestrutura do protótipo desenvolvido na primeira fase do GT-MRE.

3. Reprojetar sistema de escalonamento de usuários visando requisitos do projeto.

4. Conceber e desenvolver partes do protótipo para o modelo de servidor de laboratório de acordo com as novas tendências encontradas na literatura.

5. Testar e avaliar cada parte do protótipo, bem como a integração entre os serviços.

[4]O grupo de trabalho IEEE P1876 tem por objetivo definir métodos para armazenar e recuperar objetos de aprendizagem para os laboratórios remotos. A norma, que recebe o mesmo nome do grupo, também define como vincular objetos de aprendizagem para projetar e implementar ambientes de aprendizagem inteligentes para laboratórios remotos online. Mais informações podem ser encontradas no sistema de normas da IEEE: `https://ieee-sa.centraldesktop.com/1876public/`

6. Utilizar as recomendações para redefinição de requisitos e incrementação de funcionalidades no protótipo.

7. Validar o uso do protótipo com testes de conformidade e desempenho.

1.6 DELIMITAÇÕES DO TRABALHO

São inúmeras as possíveis funcionalidades que podem ser desenvolvidas em ambientes de aprendizagem com laboratórios remotos. Por isso, é importante destacar que este trabalho tratará apenas de tecnologias para escalonamento de usuários, servidor de laboratório remoto e modos de integrá-lo em outras aplicações. Ao longo do trabalho serão mencionados detalhes da solução desenvolvida na primeira fase do GT-MRE e sugeridas práticas para desenvolvimento de novos laboratórios, porém não é objetivo do trabalho explorar com profundidade estes assuntos.

1.7 ESTRUTURA DO TRABALHO

O texto está estruturado em quatro capítulos, incluindo este primeiro capítulo de introdução.

No capítulo 2 é apresentada uma síntese dos principais trabalhos, dentro do período de uma década, que tratam de sistema de gerenciamento para este domínio. Primeiramente, são introduzidos conceitos básicos do tema e, então, são apresentadas as funcionalidades básicas de um RLMS e três sistemas distintos. Em seguida, são descritos o modelo de laboratório como um serviço, especificação e arquitetura para dispositivos inteligentes, esforços de padronização e técnicas de reuso de software.

No capítulo 3 é descrita a plataforma de experimentação desenvolvida pelo GT-MRE com ênfase no desenvolvimento de servidores de laboratórios remotos. Primeiramente, é apresentada uma visão geral da solução e detalhes da arquitetura de software do ambiente de aprendizagem RELLE. Após, são descritos o sistema de escalonamento para usuário e suas abordagens de uso e testes que comparam o desempenho entre as duas abordagens. Posteriormente, é relatado o desenvolvimento de servidores de laboratórios remotos e interfaces com sistemas adjacentes. Por fim, a adaptação para a especificação de dispositivos inteligentes e testes de desempenho sobre a aplicação modelo são relatados.

No capítulo 4 são tecidas as considerações finais sobre o trabalho, destacando suas contribuições, dificuldades encontradas e oportunidades para trabalhos futuros.

2 LABORATÓRIOS REMOTOS

A experimentação está presente e, muitas vezes, é obrigatória em currículos de engenharia, ciências exatas e de educação científica. Embora os laboratórios presenciais ofereçam a oportunidade de experimentação com sistemas reais, ele vem acompanhado com um alto custo associado com equipamentos, espaço e pessoal técnico (GOMES; BOGOSYAN, 2009). Assim, outras alternativas surgiram e dois critérios são propostos por Auer et al. (2003) para caracterizar as diferentes modalidades:

- De acordo com a maneira com que os recursos são acessados, o laboratório pode ser remoto ou local.
- De acordo com a natureza física do laboratório, esse pode ser simulado ou real.

Em relação aos laboratórios reais, Heradio et al. (2016) define os recursos de acesso local como a combinação de laboratórios presenciais onde estudantes estão na frente de um computador conectado a equipamentos de medição e controle, enquanto os laboratórios de acesso remoto são definidos como um conjunto de instrumentos e dispositivos acessados através da Internet. O usuário opera e controla a planta real remotamente através de uma interface de experimentação.

Os laboratórios remotos permitem que seus usuários alterem parâmetros de entrada, operem instrumentos e coletem dados resultantes de um equipamento disponibilizado na sua ou outra instituição através da Internet. O sistema que suporta este tipo de interação é composto principalmente por dois nós: o servidor e o cliente.

Mesmo usando diferentes tecnologias, a maioria das soluções de laboratório remoto se baseia em uma arquitetura de software comum e simplificada semelhante à apresentada na figura 1. Em um extremo há o servidor de laboratório que consiste em um experimento e um computador e permite o controle do equipamento e conexão com uma rede de computadores. No outro extremo, há usuários que usam uma aplicação cliente, geralmente um navegador web, que oferece o controle e monitoramento remoto do experimento.

Entre o servidor de laboratório e o usuário geralmente é apresentado o RLMS, uma aplicação genérica que provê portal de acesso e funcionalidades comuns a todos os laboratórios. Um RLMS possui pelo menos uma das seguintes funcionalidades: autenticação, autorização, escalonamento de usuário para garantir o acesso exclusivo, rastreamento de usuário e ferramentas de administração.

Porém, grande parte dos laboratórios são desenvolvidos visando uma característica em específico que variam em relação à sua natureza, como laboratório de robôs (CARDOZO et al., 2010; JARA et al., 2011) e de circuitos elétricos (TAWFIK et al., 2013). Estes laboratórios são chamados por Orduña (2013) de laboratórios remotos de propósito específico (SPRL[1]) e compreendem um grande número de laboratórios presentes na literatura, assim como as tecnologias utilizadas (GRAVIER et al., 2008).

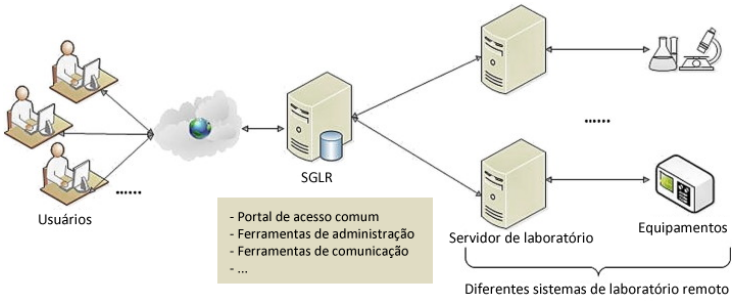

Figura 1 – Visão geral da arquitetura de um laboratório remoto. Traduzido de Lowe et al. (2016)

Esse capítulo apresenta uma revisão detalhada dos sistemas e tecnologias que dão suporte aos ambientes com laboratórios remotos existentes atualmente, focando principalmente nos aspectos de reusabilidade dos componentes de software.

2.1 SISTEMAS DE GERENCIAMENTO DE LABORATÓRIOS REMOTOS

Um componente fundamental para o desenvolvimento deste tipo de sistema é a criação de um RLMS que provê um framework comum para acessar e administrar laboratórios distintos e que oferece alguns serviços como agendamento, rastreamento de usuários e ferramentas de comunicação síncrona e assíncrona (ORDUÑA et al., 2014). Muitas pesquisas defendem que os RLMSs devem ser independentes da concepção do laboratório remoto, a fim de apoiar o maior número possível de laboratórios remotos (LOWE et al., 2016; GARCÍA-ZUBIA et al., 2009; LOWE et al., 2009).

[1] Acrônimo em inglês para Specific Purpose Remote Lab

Os RLMSs são responsáveis por gerenciar a interação entre todos os componentes e as interfaces de um sistema. Com base nos trabalhos de Maiti, Maxwell e Kist (2013), Gomes e Bogosyan (2009), um RLMS típico contém certos componentes comuns:

(a) **Controle de concorrência** - Este é um dos aspectos mais importantes deste modelo de laboratório. Como os usuários online não estão cientes da atividade de cada um dentro do sistema, as interações com hardware do experimento precisam ser coordenadas. Existem pelo menos duas estratégias de controle de concorrência utilizadas: por filas e por agendamento.

(b) **Operações sobre os equipamentos** - O laboratório é constituído de um conjunto de dispositivos ou instrumentos controlados por um computador. Nesse caso, o RLMS envia requisições e recebe dados do experimento.

(c) **Interface de usuário do experimento** - Os usuários geralmente interagem com o experimento através de um navegador web, mas também são possíveis outros clientes, como jogos e aplicativos para dispositivos móveis. As interfaces de usuário permitem o usuário observar, interagir e controlar o equipamento de teste, bem como adquirir dados e resultados.

(d) **Processamento de requisições** - Os usuários têm controle limitado dos recursos e das entradas aceitas. No processamento de requisição é realizada a validação das entradas, bem como são disparadas funções que tratam da lógica da aplicação. Este componente provê interface para integração com o cliente (UI), geralmente sobre um protocolo suportado pela maioria das aplicações cliente, como HTTP ou WebSocket.

(e) **Dados e/ou ferramentas sobre o experimento** - Qualquer informação necessária para o usuário entender, ver e analisar os dados do experimento. A transmissão de vídeo ao vivo, por exemplo, é uma ferramenta geralmente necessária para observar certas mudanças visuais.

(f) **Gerenciamento de usuário** - Neste bloco são armazenadas e manipuladas informações dos usuários, sendo que algumas destas informações são fundamentais para executar tarefas de autenticação e autorização.

Nas próximas seções serão apresentadas características comuns dos RLMSs e tecnologias utilizadas para o desenvolvimento. Após, serão analisados alguns RLMS projetados para gerenciar coleções de laboratórios remotos. Em seguida, serão discutidas novas tendências de desenvolvimento de

laboratórios remotos e ambientes de aprendizagem. Serão apresentados o paradigma de laboratórios como um serviço e de dispositivos inteligentes, bem como aspectos de reusabilidade dos componentes de software, especificações e padronização em diferentes níveis.

2.1.1 iLab Shared Architecture

O iLab Shared Architecture (ISA) é uma infraestrutura de serviço web desenvolvido pelo Massachusetts Institute of Technology (MIT) que provê um framework unificado e independente de domínio para desenvolvedores de laboratórios. Os usuários e laboratórios podem estar distribuídos globalmente em diferentes locais e conectados somente pela Internet. Os usuários utilizam os laboratórios por meio de autenticação *Single Sign On* (SSO) e uma interface administrativa padrão (HARWARD et al., 2008).

O framework ISA favorece o compartilhamento de laboratórios online, o que requer uma arquitetura que suporte o serviço do laboratório e o serviço de usuários de modo independente. O ISA diferencia as atividades de uso específico de um laboratório das tarefas que ocorrem antes, como gerenciamento de contas de usuários, autenticação e autorização, e outras tarefas após o término da sessão do laboratório, como armazenamento das configurações, resultados e preferências dos usuários. A transparência dos serviços web fazem essa tecnologia uma óbvia escolha para integrar o framework ISA de forma distribuída. No entanto, o objetivo do ISA não é prover padrões e detalhes sobre a interface de usuário do laboratório e seu controle, sendo que essas são atividades específicas de cada desenvolvedor de laboratório ou distribuidor de software.

No framework ISA são previstas três categorias de experimento:

- **Experimentos em lote** são aqueles que todo comportamento pode ser especificado antes que o experimento inicie. Os parâmetros para o teste são definidos pelo usuário e, a partir disto, o experimento é controlado pelo servidor de laboratório, retornando os resultados somente ao fim do experimento.

- **Experimentos Interativos** são aqueles no qual o usuário monitora e controla um ou mais aspectos do experimento durante sua execução.

- **Experimentos de sensores** são aqueles no qual o usuário somente monitora e analisa fluxo de dados em tempo real sem influenciar o fenômeno a ser medido.

Cada categoria de experimentos requer um conjunto diferente de serviços.

Por exemplo, em experimentos em lote, visto que não é necessário que o usuário esteja online após o início da execução do experimento, deve haver mecanismos para recuperar os resultados mais tarde. Por outro lado, em experimentos interativos necessitam escalonar usuários online enquanto o experimento ocorre. A última versão do ISA suporta duas categorias de experimentos, em lote e interativo. Para cada categoria é utilizado um middleware diferente, chamados de ISA Batch Middleware (ISABM) e ISA Interactive Middleware (ISAIM).

A arquitetura do ISABM, apresentada na figura 2, mostra como dois campi, que podem ser instituições distintas, hospedam um *Lab Server* em cada local. O *Lab Server* no ISABM é o sistema que contém toda lógica dependente das operações do laboratório e vai anexado ao equipamento final usado pelos estudantes. Além disso, cada campus possui um *Service Broker*, que é o servidor que executa o gerenciamento de usuários e permissões e o acesso ao *Lab Server*.

Os estudantes conectam-se com suas credenciais no *Service Broker* do seu campus e, logo, obtém acesso aos experimentos permitidos a esses usuários, independente da localização do laboratório. O controle de concorrência é realizado pelo *Lab Server*, que oferece um serviço web para filas que pode ser implementado em diferentes maneiras para diferentes *Lab Server* contanto que mantenha a mesma interface (HARWARD et al., 2008).

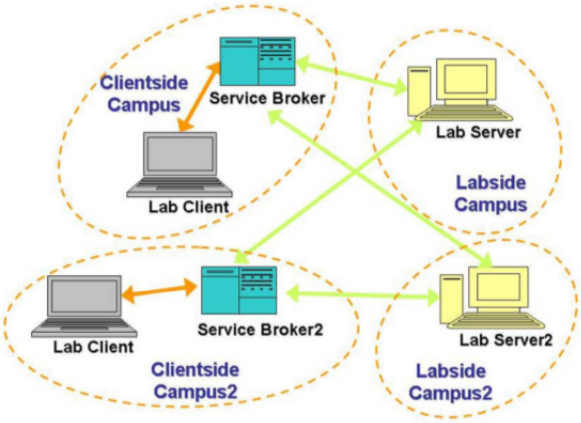

Figura 2 – Topologia da arquitetura do iLab para experimentos em lote (ISABM). Fonte: Harward et al. (2008)

Parte do ISABM foi originalmente escrito em .NET e seus serviços

adotam o protocolo SOAP. Os autores Payne e Schulz (2013) implementaram o ISABM em Java para remover a dependência das ferramentas do Windows, mantendo a compatibilidade com servidores de laboratórios existentes em SOAP. Posteriormente, Colbran e Schulz (2015) reportaram uma atualização do framework para suportar novas tecnologias de serviços web, como RESTful web services e dados formatados em JSON, além de garantir retrocompatibilidade com servidores de laboratório SOAP e um agente para gerenciar a autenticação de usuários vindos de diferentes sistemas.

A arquitetura para experimentos interativos difere bastante do ISABM porque no ISAIM a comunicação entre cliente e *Lab Server* deve ser otimizada. Na figura 3, cliente e *Lab Server* são conectados diretamente usando o protocolo escolhido pelo desenvolvedor do *Lab Server*, enquanto a comunicação com outros servidores do ISA permanece baseada em SOAP. No entanto, um novo componente, o *Experiment Storage Service*, é necessário porque nem toda informação passará pelo *Service Broker* como ocorre no ISABM. Assim, informações como relatórios de experiências anteriores e de ações do usuário são coletadas pelo *Lab Server* e submetidas ao *Storage Service*.

O *Service Broker* continua sendo responsável pela autenticação e autorização do usuário no seu próprio campus. Porém, o controle de acesso é realizado por meio de um sistema de reserva que contém dois componentes que se comunicam entre si: o *Lab-side Scheduling Service* e o *User-side Scheduling Service*. Estes dois serviços permitem maior flexibilidade para que os envolvidos (estudante, professor, proprietário do laboratório ou outros administradores de ambos lados das duas instituições) possam alterar a reserva no modelo federado.

2.1.2 Labshare Sahara

O sistema de gerenciamento Sahara foi desenvolvido pela University of Technology Sidney entre 2000 e 2005, e adotado como base para um projeto maior chamado Labshare. O Labshare foi uma iniciativa de consórcio de universidades pertencente a Rede Australiana de Tecnologia que resultou na criação do Instituto Labshare, responsável por promover, manter e hospedar os laboratórios remotos[2] (LOWE; MACHET; KOSTULSKI, 2012). O RLMS Sahara é de código-fonte aberto[3] e distribuído sob a licença BSD. A arquitetura do Sahara baseia-se principalmente em SOAP e é composta por três componentes principais, também ilustrados na figura 4.

1. Uma interface web na qual o usuário se autentica e seleciona o labo-

[2] http://www.labshare.edu.au

[3] https://github.com/sahara-labs

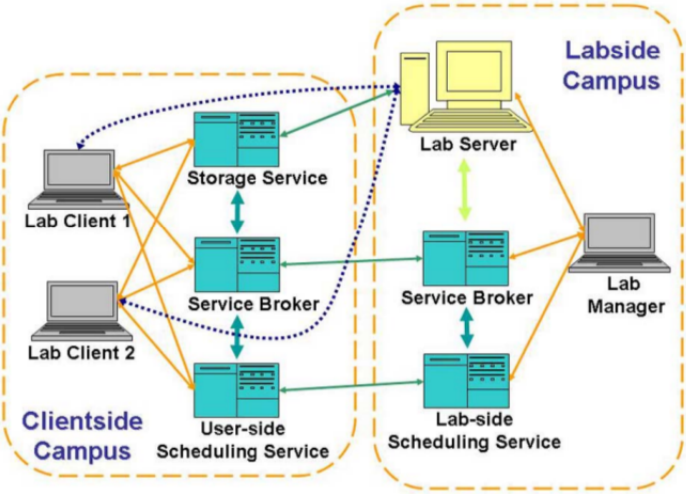

Figura 3 – Topologia da arquitetura do iLab para experimentos interativos (ISAIM). Fonte: Harward et al. (2008)

ratório em que deseja trabalhar.

2. Um servidor de escalonamento que gerencia a alocação de recursos e gerencia as sessão de usuários, bem como armazena registros de eventos e atividades.

3. Um servidor de laboratório, também chamado de *RigClient*, que é a abstração de software do laboratório e provê as operações para controle do equipamento físico.

A arquitetura de software do Sahara foi projetada para facilitar a concepção de laboratórios remotos e reduzir os custos ao criar um novo, visto que o Sahara possui protocolos e configurações mais simples para serem usados pelos criadores de laboratórios (YEUNG; LOWE; MURRAY, 2010). Lowe, Machet e Kostulski (2012) definem três tipos de experimentos conforme o tipo de controle que o *RigClient* tem sobre o experimento: controle periférico, controle direto e controle em lote.

- No controle periférico o *RigClient* não têm controle direto do hardware do experimento, pois esse geralmente é uma adaptação de um experimento que já possuia uma interface de comunicação e software para

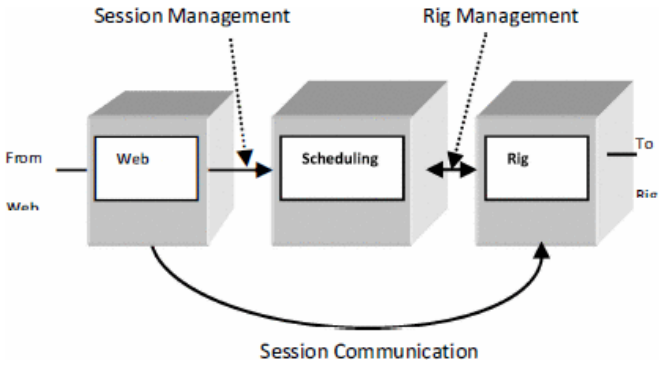

Figura 4 – Topologia da arquitetura do Sahara para experimentos interativos. Fonte: Lowe e Orou (2012)

funcionar independente. Nesta categoria estão experimentos interativos que usam LabView, por exemplo.

- No controle direto o *RigClient* é responsável por controlar o hardware do experimento, sendo mais adequado para situações com restrições de segurança ou que não há uma aplicação de controle desenvolvida.

- O controle em lote é semelhante à categoria de mesmo nome no framework ISA. O *RigClient* é responsável por carregar um arquivo de controle, verificar seu conteúdo e analisá-lo para executar a ação apropriada no experimento.

No entanto, as últimas versões do Sahara proveem ferramentas somente para laboratórios interativos, isto é, de controle periférico e direto (LOWE; OROU, 2012). A partir da versão 3, o Labshare Sahara suporta diversos mecanismos de autenticação de usuário, incluindo credenciais em banco de dados próprio, Shibboleth e LDAP.

Além disso, o Sahara passou a suportar dois métodos de escalonamento para manipular as requisições de usuário: filas e reservas. A fila garante o acesso ao recurso assim que esse estiver disponível, ou seja, os estudantes são atendidos pelo algoritmo FCFS, enquanto a reserva garante o acesso em um tempo específico a escolha do usuário.

No entanto, o uso concomitante de filas e reservas leva a um impasse quando o tempo de sessão são definidos e estáticos. Por exemplo, no caso de um usuário estar esperando na fila, ele só poderá alocar o laboratório se for possível completar uma sessão antes do prazo em que há uma reserva. Por outro lado, ainda considerando o cenário anterior, não permitir que o usuário

na fila use o laboratório significa que o recurso ficará ocioso eventualmente (LOWE; OROU, 2012).

2.1.3 WebLab-Deusto

O WebLab-Deusto é um RMLS desenvolvido pela Universidade de Deusto, Espanha, desde 2004. Este RLMS sofreu diversas modificações em termos de funcionalidades e tecnologias, principalmente adequando-se a novas tecnologias de desenvolvimento web, e atualmente encontra-se na versão 5.0. O sistema também tem código aberto distribuído sob a licença BSD 2-Clause.

O WebLab-Deusto usa arquitetura em camadas. Primeiramente, os usuários informam suas credenciais em um servidor isolado chamado *login server*. Em seguida, esse servidor valida as credenciais e solicita uma nova sessão ao *core server*, que é responsável por toda lógica de negócio. Por exemplo, ele verifica se o usuário tem autorização para um determinado laboratório, conecta-se com outros ambientes federados, rastreia as ações do usuário, etc. A maioria das interações com os experimentos ocorre através desse servidor, que irá encaminhar para o servidor próprio de cada laboratório (ORDUÑA, 2013).

Os laboratórios disponíveis no WebLab-Deusto estão espalhados pela universidade, isto porque o sistema foi construído com base nas seguintes premissas:

- Os laboratórios podem estar espalhados pela universidade e podem não ter IP público.
- O código dos experimentos não é totalmente confiável.
- A segurança física do laboratório pode ser comprometida.

Sua arquitetura é dividida em duas subarquiteturas: uma para laboratórios gerenciáveis e outra para não-gerenciáveis (ORDUÑA et al., 2011). Na arquitetura gerenciável, ilustrada na figura 5, toda comunicação entre o usuário e o laboratórios pode ser gerenciada pelo RLMS. O cliente submete um conjunto de comandos ao WebLab-Deusto Client API, que por sua vez enviará ao servidor final utilizando a API WebLab-Deusto Server. Ambas APIs estão disponíveis em várias linguagens de programação, além de suportarem múltiplas instâncias do mesmo laboratório e realizar análise dos dados de acesso.

A arquitetura de experimentos não-gerenciáveis foi criada para suportar laboratórios com interface de acesso já existentes. Apesar de o RLMS

também prover autenticação, autorização e escalonamento para essa abordagem, o usuário é direcionado para outra aplicação para usar o protocolo específico do desenvolvedor do laboratório, como o painel remoto do LabVIEW ou laboratório baseado em máquina virtual (VNC ou Remote Desktop).

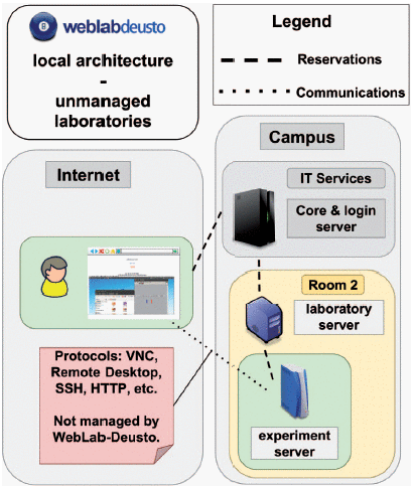

Figura 5 – Topologia da arquitetura do WebLab-Deusto para experimentos gerenciáveis. Fonte: (ORDUNA et al., 2014)

2.2 LABORATÓRIO COMO UM SERVIÇO

O potencial de impacto na aprendizagem é um dos principais indicadores de utilidade de um laboratório remoto. Segundo Corter et al. (2011), os resultados de aprendizagem dependem muito mais das instruções dadas aos estudantes e de diferentes padrões de trabalho e colaboração do que do formato de laboratório. Com esta preocupação, diversos grupos de pesquisa voltaram sua atenção ao formato em que o conteúdo de laboratórios remotos é entregue e seu impacto pedagógico.

Uma das questões relacionadas à experiência de aprendizagem é a integração entre sistemas educacionais heterogêneos, serviços e objetos de aprendizagem. O compartilhamento de laboratórios remotos entre instituições é fundamental para tornar viável a construção de ambientes de aprendizagem com laboratórios remotos, principalmente por aumentar a disponibilidade e reduzir custos. Em resposta a estas necessidades, o conceito de laboratório como um serviço (ou *Lab as a Service* - LaaS) é defendido por Tawfik et

al. (2014) como uma maneira organizada para implementar e compartilhar laboratórios.

O modelo deriva do conceito *Anything as a Service* (XaaS), que refere-se à grande quantidade e diversidade de serviços disponíveis na Internet, inclusive para serviços de dispositivos que são alcançáveis através de diferentes clientes. A abordagem de desenvolvimento a partir de componentes modulares do LaaS busca implementar serviços de um laboratório como um conjunto de serviços fracamente acoplados para serem consumidos a um alto nível de abstração. Este modelo implica em entregar todas funções e componentes de um laboratório em um arquivo de descrição de serviço como um conjunto de serviços abstratos.

O LaaS segue a arquitetura SOA e cumpre seus requisitos básicos, como independência de plataforma, sistema operacional e de linguagem de programação, descrição do serviço em síntaxe clara e não-ambígua, e acesso através de protocolos padronizados. Os objetivos desse modelo podem ser definidos como:

- Definir uma maneira organizada para compartilhar laboratórios.
- Permitir a integração de laboratórios em qualquer aplicação independente da tecnologia adotada e do seu acoplamento com outros serviços.
- Facilitar manutenção e reusabilidade.
- Permitir a intercambiabilidade de componentes entre provedor e consumidor.
- Servir como princípio para um padrão de projeto global para desenvolvimento e implementação de laboratórios remotos.

O LaaS define a relação entre provedores de laboratório (quem provê arquivos de descrição de serviço), repositório de serviços ou *market place*[4] (local onde os arquivos de descrição de serviço são indexados) e consumidores (quem desenvolve uma aplicação sobre os serviços fornecidos). Da perspectiva de orientação a serviço, os arquivos de descrição de serviço devem ser registrados em um servidor intermediário e entregue aos usuários sob demanda. A sequência de interação entre provedor, repositório e consumidores é descrita como:

1. Os provedores de diferentes instituições exportam seus laboratórios em um *market place* e atribuem ontologias ao seu laboratório dentro do *market place*.

[4]Local para venda de serviços de terceiros

2. O *market place* age como um repositório global de laboratórios.

3. Consumidor ou estudantes buscam por laboratórios e os importam no ambiente de aprendizagem.

4. Os laboratórios são importados como serviços com suas APIs e política de acesso, se disponível, são disponibilizadas.

5. Consumidores usam o serviço importado e comunicam-se diretamente com o equipamento.

As tentativas de usar tecnologias de serviços web, até então adotadas somente para aplicações corporativas, para sistemas de laboratórios remotos iniciaram com o trabalho de Saliah-Hassane et al. (2005). Os autores descreveram o serviço de um equipamento industrial da National Instruments com WSDL e registraram em um serviço de diretório para ser alocado e consumido através do protocolo SOAP. Outros exemplos de controle de instrumentos online usando serviços web SOAP também são encontrados nos trabalhos de Capua, Liccardo e Morello (2005) e Tröger et al. (2008). Em seguida, Ngolo et al. (2009) descreveram uma arquitetura baseada em serviços web REST e clientes em AJAX, seguindo a tendência de tecnologias web.

Uma abordagem distinta chamada de paradigma de dispositivos inteligentes foi descrita por Salzmann e Gillet (2013) para adicionar autonomia aos laboratórios remotos no lado do servidor e independência do cliente. Este paradigma foi concebido para ser base da infraestrutura do projeto Go-Lab[5]. Além disso, atualmente este paradigma é recomendação do grupo de trabalho IEEE P1876 para implementação do modelo LaaS.

O estudo de caso de Tawfik et al. (2014) demonstrou um protótipo modular de laboratório remoto entregue de acordo com o modelo LaaS. A ideia por trás da modularização é facilitar a manutenção, reusabilidade e intercambiabilidade de componentes do laboratório. Um dos objetivos dos autores foi reunir embasamento teórico e técnico para propor um modelo padrão aceitável para que desenvolvedores e provedores de laboratórios pudessem adotar.

[5]O projeto Go-Lab promove o amplo uso laboratórios online de ciência na educação básica e secundária. Seu principal objetivo é dar suporte pedagógico e de infraestrutura ao portal Go-Lab para professores criarem espaços de aprendizagem investigativa. A documentação sobre infraestrutura está disponível em: http://www.go-lab-project.eu/deliverables

2.2.1 Dispositivos Inteligentes

Na área de sensores, os transdutores inteligentes[6] ampliam funcionalidades de sensores estáticos com a adição de informações. Um simples sensor, em vez de retornar somente um valor de tensão, pode retornar a frequência de amostragem, faixa de valores ou informações de calibração, por exemplo. O mesmo conceito pode ser extendido para dispositivos de IoT, dando origem ao paradigma de dispositivos inteligentes.

Thompson (2005) define um dispositivo inteligente como um dispositivo que apresenta uma ou todas seguintes capacidades: comunicação, identidade e tipo, memória e rastreamento de status, sensoriamento e atuação, e raciocínio e aprendizagem. Uma extensão desta proposta é apresentada por Salzmann e Gillet (2013) para suportar dispositivos mais complexos como laboratórios remotos. Assim, o paradigma é usado como framework arquitetural para conectar um experimento remoto online e propõe separar os dois componentes que geralmente são fortemente acoplados: a aplicação cliente e servidor de laboratórios remotos. Este desacoplamento é atingido através da adição de autonomia e inteligência e de conceitos da computação orientada a serviço à aplicação do servidor de laboratório (GOVAERTS, 2014).

Em geral, o paradigma recomenda funcionalidades internas de um dispositivo autônomo ideal e interfaces bem definidas para acessá-las. A primeira especificação de dispositivos inteligentes para laboratórios remotos foi desenvolvida pelos pesquisadores participantes do projeto Go-Lab. Essa especificação é parte fundamental na infraestrutura da plataforma para garantir interoperabilidade entre aplicações cliente, neste caso chamada de plataforma de aprendizagem investigativa, e diferentes fornecedores de laboratórios online. Os metadados definidos na especificação são comuns para serem compartilhados por todos dispositivos, permitindo reuso de rotinas em aplicações cliente.

2.2.1.1 Arquitetura para dispositivos inteligentes

Como mencionado anteriormente, os dispositivos inteligentes foram especificados por participantes do projeto Go-Lab visando resolver problemas devido à falta de metadados e serviços comuns, definição de responsabilidades de cada componente e reusabilidade das aplicações. O cenário deste projeto vai além da especificação dos dispositivos inteligentes, pois também

[6] Refere-se à família de normas IEEE 1451 que define *datasheets* eletrônicos para descrever transdutores e interfaces de comunicação, permitindo implementação de características *plug-and-play* a sensores e atuadores.

implementa serviços que apoiam um ambiente de aprendizagem investigativa, como indexação e motor de busca de laboratórios, análise de aprendizagem e cliente para visualização de dados. Assim, as interfaces bem definidas dos dispositivos inteligentes asseguram que qualquer aplicativo cliente e serviço pode se comunicar com qualquer dispositivo inteligente.

Além de permitir a interação entre usuário e laboratório, os dispositivos inteligentes podem ser acessados pelas plataformas e serviços do Go-Lab para: publicar laboratório no repositório[7], recuperar metadados, rastrear atividade do usuário e reservar um laboratório, por exemplo (GOVAERTS, 2014). A figura 6 apresenta uma arquitetura básica com alguns exemplos de interação com o dispositivo inteligente através de um conjunto de interfaces obrigatórias e opcionais. Além disso, a figura 6 não apresenta a comunicação entre o dispositivo e o equipamento do laboratório remoto porque a especificação não a define. O dispositivo inteligente apenas especifica protocolos e formato de dados de interfaces (isto é, metadados, clientes, sensores, atuadores e logs) que serão detalhados na seção 2.2.1.2.

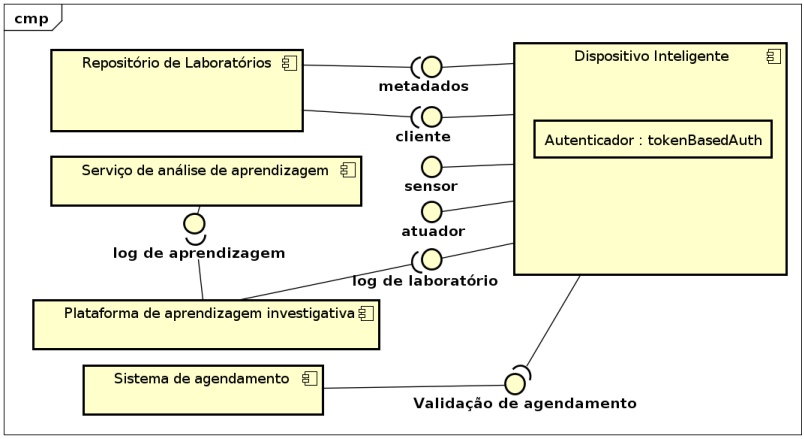

Figura 6 – Diagrama de composição da infraestrutura do projeto Go-Lab. Traduzido de: (SALZMANN; GILLET, 2013)

Como o dispositivo inteligente é indiferente à interface de usuário, não é responsabilidade do dispositivo prover uma interface para controle do laboratório. No entanto, ele poderá fornecer uma interface de usuário simples e/ou retornar via serviço web uma lista com endereço e descrição de clientes disponíveis. Uma proposta relacionada à organização de interfaces de usuário

[7] http://www.golabz.eu

utiliza blocos dinâmicos de conteúdo, chamados de *widgets*, para acesso aos dispositivos. A vantagem desta abordagem está na flexibilidade e reusabilidade de componentes da interface gráfica para ambientes virtuais e aplicativos baseados em tecnologias web. Neste caso, a página é composta por *widgets* que podem ser organizados de diferentes formas (GARCÍA et al., 2013).

2.2.1.2 Especificação de metadados e serviços

A especificação de dispositivos inteligentes apresentada por Govaerts (2014) define a comunicação e interfaces entre o cliente e o servidor de laboratórios remotos. Ela possui seções que definem terminologia, protocolo de transferência de dados, serviços obrigatórios e opcionais de recuperação de metadados, serviço de dados e melhores práticas. Os termos recorrentes em trabalhos sobre experimentação remota, como tipos mecanismo de controle de concorrência, de sensores e atuadores, foram elencados para serem utilizados nos serviços de dados e de recuperação de metadados.

Como as aplicações Web são tendência para acesso aos laboratórios remotos, a especificação considera somente protocolos padrões da Web. Dos candidatos atuais, HTTP e WebSockets, o WebSocket é considerado o mais adequado porque os experimentos remotos geralmente precisam transferir dados de maneira assíncrona, sendo que soluções com HTTP, como HTTP *polling*, seriam ineficientes. Considerando outras vantagens como disponibilidade na maioria dos navegadores e, geralmente, menor esforço para programação, o WebSocket foi adotado como protocolo padrão da especificação. Uma exceção é feita para os serviços de metadados, em que foi escolhido HTTP para facilitar a recuperação de informação.

O arquivo de descrição de serviços apresenta informações básicas do laboratório e de integração com serviços externos, além de definir o formato de dados de requisição e resposta. Neste arquivo é descrita a forma de acesso aos serviços, que é obrigatório para garantir a interoperabilidade prevista na especificação e suficiente para automaticamente definir interface de usuário e de comunicação com o servidor do laboratório. A linguagem de descrição de serviços Web adotada foi a OpenAPI[8]. A OpenAPI é uma linguagem de descrição baseada em JSON facilmente adaptável para uso com WebSockets, sendo usada nesta especificação não somente para serviços de recuperação de

[8] A especificação OpenAPI, originalmente chamada de Swagger, é uma especificação para arquivos de interface legíveis por máquinas para descrever, produzir, consumir e visualizar serviços web RESTful. A OpenAPI é agnóstica em relação à linguagem e possui um motor orientado a template para geração de documentação e esqueleto de clientes e servidores em diferentes linguagens a partir de uma especifição de serviço OpenAPI. As ferramentas são de código aberto e estão disponíveis em: `https://github.com/OAI/OpenAPI-Specification`

metadados, mas também de dados de sensores e atuadores via WebSockets.

Os metadados gerais dos dispositivos inteligentes são apresentados na tabela 1. Os metadados da API referem-se à descrição serviços de recuperação de metadados de sensores e atuadores disponíveis, sendo separados em operações identificadas por um nome de método, como ”getSensorMetadata”e ”getActuatorMetadata”, podendo ser acessados via HTTP ou WebSocket. Os serviços de dados também são identificados por um nome de método, como ”getSensorData”e ”sendActuatorData”, mas são acessados somente via WebSocket. Além dos serviços obrigatórios, a especificação pode incluir a recuperação de metadados de aplicações clientes disponíveis, acesso a logs e modelos matemáticos ou gráficos do laboratório. Para exemplificar a sequência de interação entre o cliente e o dispositivo inteligente através dos serviços descritos na especificação, é apresentado um diagrama de sequência na figura 7.

O repositório do projeto Go-Lab no Github[9] contém a especificação do formato de metadados para o Go-Lab Smart Device usando o Swagger para definir as APIs do dispositivo inteligente.

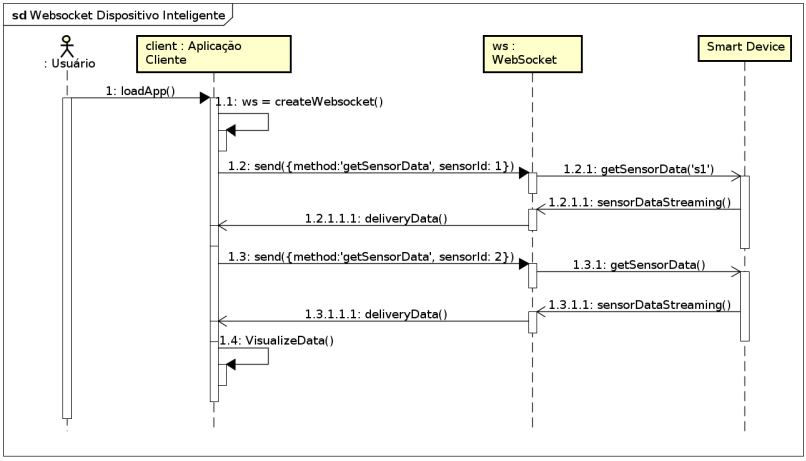

Figura 7 – Interação entre cliente e dispositivo inteligente. Traduzido e adaptado de: (GOVAERTS, 2014)

[9] <https://github.com/go-lab/smart-device-metadata>

Tabela 1 – Especificação de metadados gerais de dispositivo inteligente. Fonte: Govaerts (2014)

Tipo	Descrição
Informações	Campos contém informações sobre o laboratório, como nome, descrição, proprietário, informações de contato e licença.
Metadados da API	Contém caminho para construção de URLs, nome de operações e seus parâmetros, referência aos formatos de requisição e/ou saída e informação de autorização para acesso aos serviços no dispositivo inteligente.
Metadados da OpenAPI	Contém versão da especificação OpenAPI e versão da API.
Metadados de autorização	Contém mecanismos de autenticação e/ou autorização suportados pelo dispositivo. As ferramentas de geração de esqueletos da OpenAPI suportam mecanismos de autenticação e autorização como OAuth.
Metadados de acesso concorrente	Provê a descrição de diferentes mecanismos de concorrência para que o proprietário decida o tipo apropriado para seu laboratório, e, assim, a aplicação cliente pode manipular diferentes modelos. São definidos dois tipos de papéis, fixo e dinâmico, e três mecanismos usados para alternar papéis, competição, fila e interruptor.
Formato de requisições e respostas	Contém a lista de todos modelos em esquema JSON usados para requisição e resposta de serviços.

2.2.2 Gateway4Labs

A integração de laboratórios remotos em ferramentas de aprendizagem têm sido amplamente discutida na literatura. Embora essa integração seja um tópico importante para a área, a maioria dos laboratórios encontrados na literatura ainda são *ad-hoc*, ou seja, são desenvolvidos e usados no LMS de uma instituição e, possivelmente, não irá funcionar no LMS de outra instituição

(SANCRISTOBAL et al., 2010; ORDUNA et al., 2014). A abordagem de LaaS e a especificação de dispositivos inteligentes são adequadas para o desenvolvimento de novos laboratórios. No entanto, laboratórios online não concebidos de acordo com a especificação necessitariam de aplicações intermediárias para habilitar o acesso a uma infraestrutura que provê serviço de repositório e de análise de aprendizagem, por exemplo.

A exemplo da infraestrutura Go-Lab, são possíveis três formas de conectar sistemas legado à plataforma. A primeira consiste em redesenhar o sistema de acordo com as especificações de dispositivos inteligentes. A segunda abordagem incorpora a aplicação web do laboratório como um iFrame na plataforma, nesse caso nenhuma comunicação entre o laboratório e a plataforma é necessária. A última abordagem é a integração via Smart Gateway, que assegura um grau maior de compatibilidade em relação à segunda abordagem (HALIMI; GOVAERTS, 2015).

O Smart Gateway é um *middleware* utilizado entre o sistema de laboratório legado e a infraestrutura do Go-Lab para maior compartibilidade e conformidade com a especificação de dispositivos inteligentes. Ao optar por usar Smart Gateway, o proprietário do laboratório não precisa implementar a integração com o portal Go-Lab para o sistema de agendamento, serviço de metadados ou qualquer outro serviço definido no Go-Lab. Esta aplicação é genérica e flexível para suportar diferentes RLMSs e, por essa razão, ele deve prover diferentes níveis de integração.

O Smart Gateway é composto por dois componentes, o gateway4labs (G4L) e o tradutor de protocolo. O gateway4labs[10] é um componente que suporta API OpenSocial e um sistema de *plugins* que possibilita o desenvolvimento de *plugins* customizados para realizar requisições ao laboratório remoto. Este *plugin* apenas direciona o usuário da plataforma para o laboratório, realizando autenticação com o sistema legado de laboratório, por exemplo. Já o tradutor de protocolos é um componente independente e opcional que reempacota a comunicação do sistema legado e as expõe como serviços compatíveis com a especificação dos dispositivos inteligentes, usando WebSockets e JSON, por exemplo.

A atual implementação Smart Gateway define quatro níveis de *plugins*:

- *Plugin* iFrame - provê metadados e reserva apenas na plataforma.
- *Plugin* simples - possibilita criação de interface de usuário e metadados customizados
- *Plugin* completo - inclui protocolos de autenticação, reserva e de recuperação

[10] https://github.com/gateway4labs

de dados.

- *Plugin* completo com tradutor de protocolo - fornece todas funcionalidades dos dispositivos inteligentes.

Assim, o Smart Gateway oferece aos proprietários de laboratório alternativas para manterem seu sistema legado através de adaptações por meio de *plugin* no gateway4labs e, opcionalmente, pela implementação de tradutor de protocolo atingir total nível de integração e adequações às especificações de dispositivos inteligentes.

2.2.3 Padronização e trabalhos em progresso

Os laboratórios onlines tornam-se cada vez mais populares e um padrão que estabeleça o relacionamento entre todos componentes facilita o projeto e implementação de ambientes e atividades online dirigidas pela pedagogia. O padrão IEEE P1876 pretende facilitar o projeto, implementação e uso destes laboratórios como objetos de aprendizagem inteligentes e sua integração em ambientes de aprendizagem e repositórios de objetos de aprendizagem. O trabalho de padronização de Objetos de Aprendizagem Inteligentes em Rede para Laboratórios Online está em fase inicial, porém já foram definidos três níveis: um nível pedagógico, um nível de serviço e um nível de protocolo de comunicação.

O nível pedagógico descreve como empacotar os recursos de uma forma padronizada e como habilitar sua integração em ambientes de aprendizagem (por exemplo, VLE, PLEs, plataformas MOOC ou plataformas de mídia social). O nível de serviço padroniza como os clientes se comunicam com um laboratório remoto. Finalmente, o nível do protocolo de comunicação padroniza a forma como todos os serviços e plataformas que suportam o uso de laboratórios remotos podem ser interoperáveis.

A figura 8 ilustra os níveis de implementação de um laboratório online como objetos de aprendizagem inteligentes considerados na padronização. O primeiro nível padroniza um laboratório online como um serviço (LaaS) que pode ser personalizado no segundo nível. O segundo nível padroniza o laboratório online como um Recurso Educacional Aberto (LaaO), o qual pode ser integrado em ambientes de aprendizagem e hospedados em repositórios de objetos de aprendizagem.

A fim de encorajar a adoção da especificação de dispositivos inteligentes, pesquisadores do projeto Go-Lab estão envolvidos no grupo de trabalho P1876 e têm seu modelo aceito como padrão inicial para o nível de serviço e de protocolo de comunicação. Além disso, os serviços de dispositivos in-

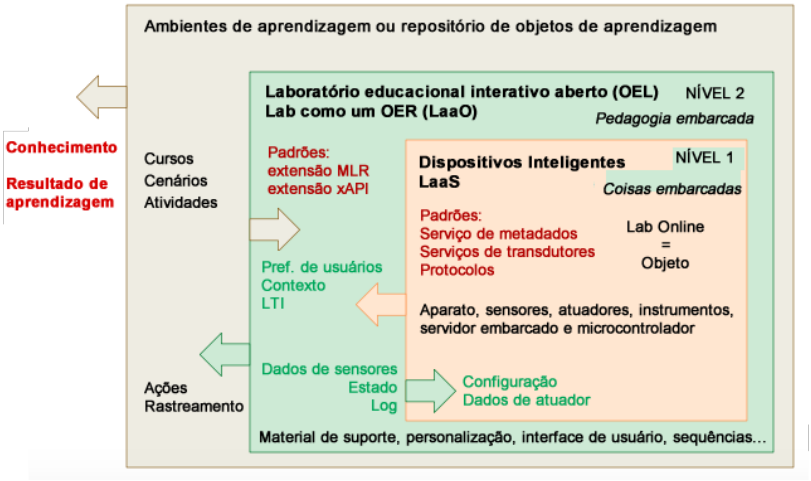

Figura 8 – Diagrama de composição da infraestrutura do projeto Go-Lab. Traduzido de: (IEEE, 2016)

teligentes podem permitir interoperabilidade para uso das mesmas funcionalidades em diferentes plataformas, como sistema de reservas ou serviços de análise de aprendizagem. A camada de abstração fornecida pela especificação de dispositivos inteligente foi adotada como base para desenvolvimento de uma especificação final a ser elaborada.

2.2.4 Reuso de software em laboratórios remotos

O reuso sistemático de software reduz custos com projeto, acelera o desenvolvimento de projetos. Sommerville (2004) divide o reuso de software em três categorias: a primeira consiste no reuso integral, seja por incorporação ou desenvolvimento de uma família de aplicações, a seguinte refere-se ao reuso de componentes de um subsistema e, a última refere-se a objetos bem-definidos e reuso de funções. As técnicas de reuso apresentadas na tabela 2 têm sido desenvolvidas nas duas últimas décadas e são fundamentais para resolver problemas de reusabilidade existentes no domínio de laboratórios online.

Um sistema de laboratório remoto pode usar mais de uma dessas técnicas quando um novo laboratório é projetado. Apesar de os RLMSs descritos nas seções anteriores serem pioneiros na reutilização, os sistemas podem não re-

Tabela 2 – Técnicas de reuso. Fonte: Sommerville (2004)

Abordagem	**Descrição**
Padrões de projetos	Abstrações genéricas que ocorrem em aplicações representadas como padrões para projetos que possuem interações e objetos abstratos e concretos
Desenvolvimento baseado em componentes	Sistemas desenvolvidos a partir da integração de componentes (coleção de objetos) que compõe um padrão de componente modelo.
Frameworks	Coleção de classes abstratas e concretas que podem ser adaptadas e estendidas para criar um aplicação.
Encapsulamento	Sistemas legados podem ser encapsulados definindo um conjunto de interfaces e provendo acesso ao sistema por esta interface.
Orientação a serviço	Sistemas são desenvolvidos pela combinação de serviços compartilhados que podem, inclusive, ser fornecidos externamente.
Linha de produção	Um tipo de aplicação é generalizado em cima de uma arquitetura comum e, assim, pode ser adaptada em diferentes maneiras para clientes diferentes.
Biblioteca da programas	Bibliotecas de classes e funções de uso comum são implementadas e suas abstrações são disponilizadas para reuso.
Geradores de programas	Um sistema gerador incorpora o conhecimento de um tipo particular de aplicação e pode gerar sistemas ou partes de sistemas para aquele domínio.
Orientação a aspecto	Componentes compartilhados são entrelaçados em uma aplicação em diferentes locais quando um programa é compilado.

fletir em todas possibilidades e resolver problemas de outros desenvolvedores (GRAVIER et al., 2008).

O paradigma de dispositivos inteligentes, como mencionado nas seções anteriores, simplifica a tarefa de desenvolver e implantas laboratórios, elimi-

nando o esforço associado à construção de uma nova aplicação. Além do paradigma de dispositivos inteligentes, há templates de software que permite que os proprietários de laboratórios tenham ponto de partida para seu trabalho. O template de software apresentado por Halimi, Salzmann e Gillet (2016) foi escrito com LabVIEW e pode ser implantado em qualquer sistema que suporta LabVIEW apenas adaptando o código de interface de hardware.

3 DESENVOLVIMENTO

O GT-MRE desenvolve laboratórios remotos e recursos educacionais abertos para disciplinas STEM. Mesmo antes da formação do grupo de trabalho, o RExLab já concentrava suas pesquisas no ganho pedagógico e na infraestrutura para a experimentação remota, promovendo o uso de laboratórios públicos, sem custo e sem necessidade de registro.

Professores e alunos eventualmente irão requerer o acesso a um laboratório através do LMS de sua instituição ou de um portal que reúne diversos laboratórios online. Assim, é necessário prever a integração com outros ambientes de aprendizagem e a abordagem de serviço web é um candidato a proporcionar fácil integração de serviços de um laboratório em qualquer plataforma.

No GT-MRE foi adotada uma abordagem distribuída e modular, principalmente focando em serviços web hospedados em computadores com capacidades limitadas e de baixo custo, como SBCs, e disponibilizados em ambiente de aprendizagem. Deste modo, cada laboratório é um dispositivo independente conectado à Internet e que provê acesso a sensores e atuadores.

Neste capítulo serão detalhados os requisitos e processos de desenvolvimento e implantação deste serviço. Primeiramente será apresentado o ambiente de aprendizagem com laboratórios remotos, suas funcionalidades, visão dos seus componentes e características responsáveis pela integração com serviços externos. Em seguida será detalhada a abordagem utilizada para criação dos laboratórios remotos, comparações com o outros trabalhos e testes de conformidade e desempenho.

3.1 AMBIENTE DE APRENDIZAGEM COM LABORATÓRIOS REMOTOS

O serviço proposto pelo GT-MRE visa desenvolver e implantar uma plataforma que integre ambiente virtual de ensino, conteúdos didáticos abertos online e interação com experimentos remotos. Os objetivos deste serviço compreendem a ampliação do acesso à laboratórios e o estímulo à integração da tecnologia para professores e estudantes da Educação Básica e Superior. A fim de alcançar esse objetivo, foi proposta a construção de oito experimentos e uma plataforma para manipulação dos laboratórios.

A parte central da solução desenvolvida pelo GT-MRE é a aplicação chamada RELLE, sigla para *REmote Lab Learning Environment*, que consiste em um ambiente de aprendizagem e também repositório de laboratórios.

O papel do RELLE é promover as atividades laboratoriais que podem ser acessadas na sua plataforma ou incorporadas em outro LMS.

Atualmente, existem 10 diferentes laboratórios desenvolvidos pelo GTMRE na coleção do RELLE. Os experimentos disponíveis[1] são nomeados como: (a) Painel Elétrico CC, (b) Painel Elétrico CA, (c) Condução de Calor em Barras Metálicas, (d) Propagação de Calor, (e) Microscópio Remoto, (f) Ambiente de desenvolvimento em Arduino, (g) Plano Inclinado, (h) Banco Óptico, (i) Conversão de Energia Luminosa em Elétrica e (g) Disco de Newton. Neste conjunto, os laboratórios *(a)*, *(f)* e *(g)* possuem duplicação de instância. Além disso, o ambiente virtual permite incluir laboratórios de outros desenvolvedores, neste caso encontram-se cadastrados os seguintes laboratórios: Experimento de Thomson[2], Observando a Água[3] e Microscópio Remoto LTE[3].

Visto que cada laboratório possui recursos que devem ser acessados apenas por um usuário por vez, um sistema de escalonamento de usuários é necessário. Neste caso, foram implementados dois modelos de escalonamento, um externo e outro interno ao servidor de laboratório. Além disso, todos os experimentos utilizam a mesma arquitetura básica. Na maioria dos experimentos está sendo utilizada uma placa de aquisição e controle que é responsável por receber e enviar sinais de sensores e atuadores e transmitir os dados de forma estruturada ao servidor de laboratório.

3.1.1 Arquitetura de software

A função principal do RELLE é: (a) prover interfaces de usuário que consomem serviços web dos laboratórios cadastrados e (b) gerenciar conteúdos educacionais relacionados aos laboratórios, como manuais, sequências didáticas e simulações. O que o difere de outro ambiente de aprendizagem com experimentos remotos existente é que seu conteúdo e interface de cada laboratório podem ser modicadas pelo criador de laboratório quando ele quiser, assim como outro sistema de gerenciamento de conteúdo.

As funções do RELLE são específicas para cada tipo de usuário. O papel de administrador de laboratório permite acesso a um painel de controle para facilmente editar e gerenciar aplicação cliente . O papel de administrador do sistema permite a criação, atualização e exclusão de aplicações cliente no banco de dados, além de permitir adicionar usuários e atribuir-lhes permissão,

[1] http//relle.ufsc.br/labs

[2] Experimento em desenvolvimento em parceria com o Núcleo de Pesquisa em Tecnologias Cognitivas da Universidade Federal de Uberlândia.

[3] Experimento desenvolvido pelo Laboratório de Tecnologias Educacionais da Universidade Estadual de Campinas.

acessar logs e análise de logs. O ambiente serve como suporte para professores e alunos visualizarem, controlarem, receberem e exportarem dados da experiência, além de acessarem material didático.

Na figura 9 é ilustrada resumidamente a relação entre o RELLE, as interfaces de usuário, servidor de laboratório e serviço de escalonamento de usuário. O RELLE segue o padrão de projetos *Modelo-Visão-Controle* provido pelo framework Laravel, que é usado como base para agilizar o processo de desenvolvimento e manutenção. Dentro do conjunto de visões encontram-se as aplicações clientes para acesso aos laboratórios remotos, cada um composto por arquivos HTML, CSS, Javascript e conteúdo multimídia. Esses clientes são hospedados no servidor do RELLE e podem ser acessados por navegadores ou aplicativos para dispositivos móveis.

O acesso aos laboratórios através do RELLE inicia em uma página web comum a todos laboratórios, na qual são apresentadas informações sobre o laboratório. Nesta página o browser do cliente comunica-se com o serviço de escalonamento (API FCFS) para que os arquivos do cliente sejam carregados dinamicamente para cada usuário que obtém acesso exclusivo. Após carregar a aplicação cliente, a comunicação com o laboratório estará completamente sob responsabilidade da aplicação cliente e, assim, pode ser usado o protocolo, estruturas de dados e métodos desejados pelo desenvolvedor do servidor de laboratório.

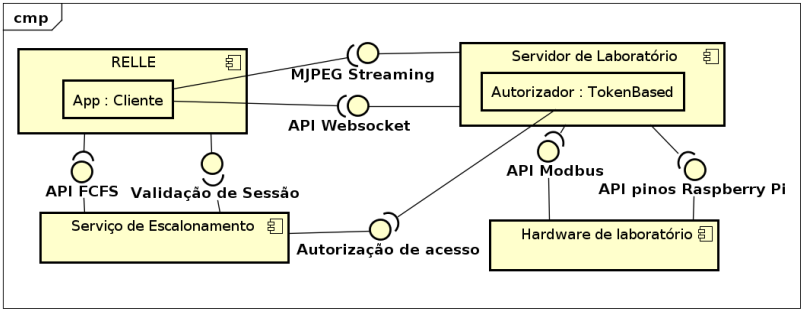

Figura 9 – Diagrama de componentes da arquitetura do RELLE. Fonte: Autor.

Em um primeiro estágio foram implementadas 11 interfaces de usuário[4], com elementos gráficos e comunicação com servidor de laboratório customizadas para cada uma. Foram utilizadas algumas bibliotecas comuns para os

[4]As aplicações cliente para os laboratórios do GT-MRE e de outros desenvolvedores encontram-se em repositório público no GitHub: `https://github.com`.

experimentos do GT-MRE, como o Socket.io[5] e i18next[6].

Como não foram utilizadas estratégias de reuso de software neste estágio, todas aplicações cliente deveriam conhecer previamente as funções oferecidas pelos laboratórios. O acoplamento forte, falta de flexibilidade e possibilidade de reutilizar código surgem como pontos negativos quando a aplicação do lado servidor e cliente são implementadas em conjunto.

Na seção 3.2.5 será mostrado como a descrição de serviços é utilizada para separar complementamente aplicação cliente e servidor e facilitar o reuso de código.

3.1.2 Escalonamento de usuários

Como discutido no capítulo anterior, o controle de concorrência está presente e é requerido em quases todos laboratórios remotos. O mecanismo utilizado pode variar dependendo do modelo de interação entre usuário e laboratório e entre os próprios usuários. Além dos diferentes papéis possíveis para os usuários (controlador, observador e interruptor), também podem ser encontradas três formas de alocação de recurso, através de agendamento, fila e mesmo um modelo híbrido entre agendamento e fila.

Resgatando a terminologia usada no iLabs, os experimentos do GT-MRE são em sua maioria do tipo interativo e, por isso, necessitam de alocação de recurso no momento de uso. Além disso, há a restrição de projeto discutida anteriormente que preza pela facilidade de uso, isto é, acesso direto, sem registro e, preferencialmente, sem reserva prévia. O escopo do sistema de escalonamento não inclui a autenticação e autorização em base de usuários, porém são definidos os protocolos de transferência de dados, estruturas de dados e operações que são consideradas fundamentais para uso de fila nos laboratórios.

A implementação do sistema de escalonamento compreende aplicações de cliente e servidor que se comunicam através de um protocolo de mensagens assíncrono e baseado em eventos. Para o lado do servidor foi desenvolvido um módulo genérico para Node.js para escalonar as sessões de usuários conectados e autorizar as operações de usuários.

Do lado do cliente, a aplicação de controle do laboratório incorpora a biblioteca responsável pela transição de estados mostrada na figura 10. Como o módulo para servidor é genérico, ele pode ser incluído na aplicação de

[5]Biblioteca Javascript para aplicações web de tempo real. Documentação disponível em `http://socket.io/`.

[6]Biblioteca Javascript para internacionalização, mas também disponível para outras linguagens de programação. Documentação em: `http://i18next.com/`.

servidor de laboratório ou usado para construir uma aplicação autônoma que gerencia múltiplos laboratórios, inclusive com replicação de instâncias.

Na seção 3.1.2.1 e 3.1.2.2 serão apresentadas duas abordagens de controle de concorrência por fila em que um único usuário tem acesso exclusivo ao recurso laboratorial.

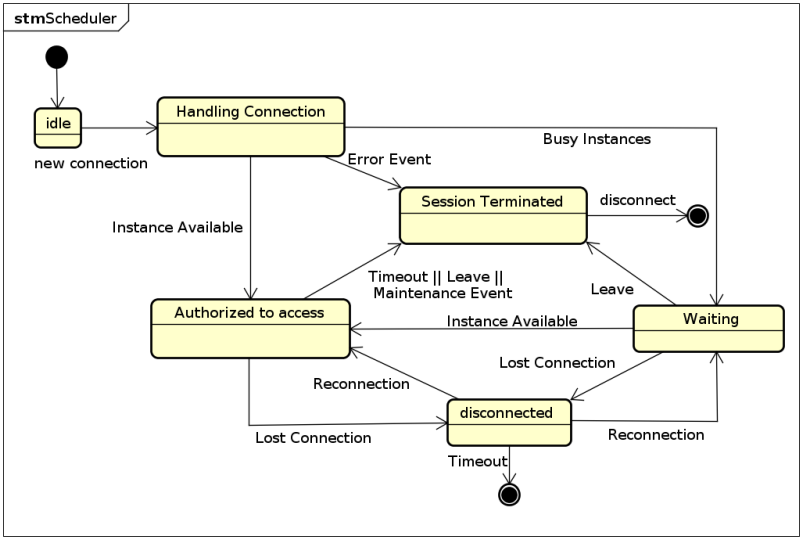

Figura 10 – Diagrama estados do sistema de escalonamento na visão do cliente. Fonte: Autor.

3.1.2.1 Externo ao servidor de laboratório

Durante a segunda fase do GT-MRE, que compreendeu o período de janeiro a dezembro de 2016, foram replicados 3 experimentos, sendo eles o Painel CC, Plano Inclinado e Laboratório de Arduino. Após a replicação cada instâncias dos experimentos citados foram cadastrados no RELLE em página diferentes e, consequentemente, com filas diferentes.

Esta réplica de página de entrada e fila para acesso é uma deficiência em termos de usabilidade. Para resolver este problema foram realizadas mudanças significativas no sistema de escalonamento desenvolvido em 2015 e descrito no trabalho de Simao (2016).

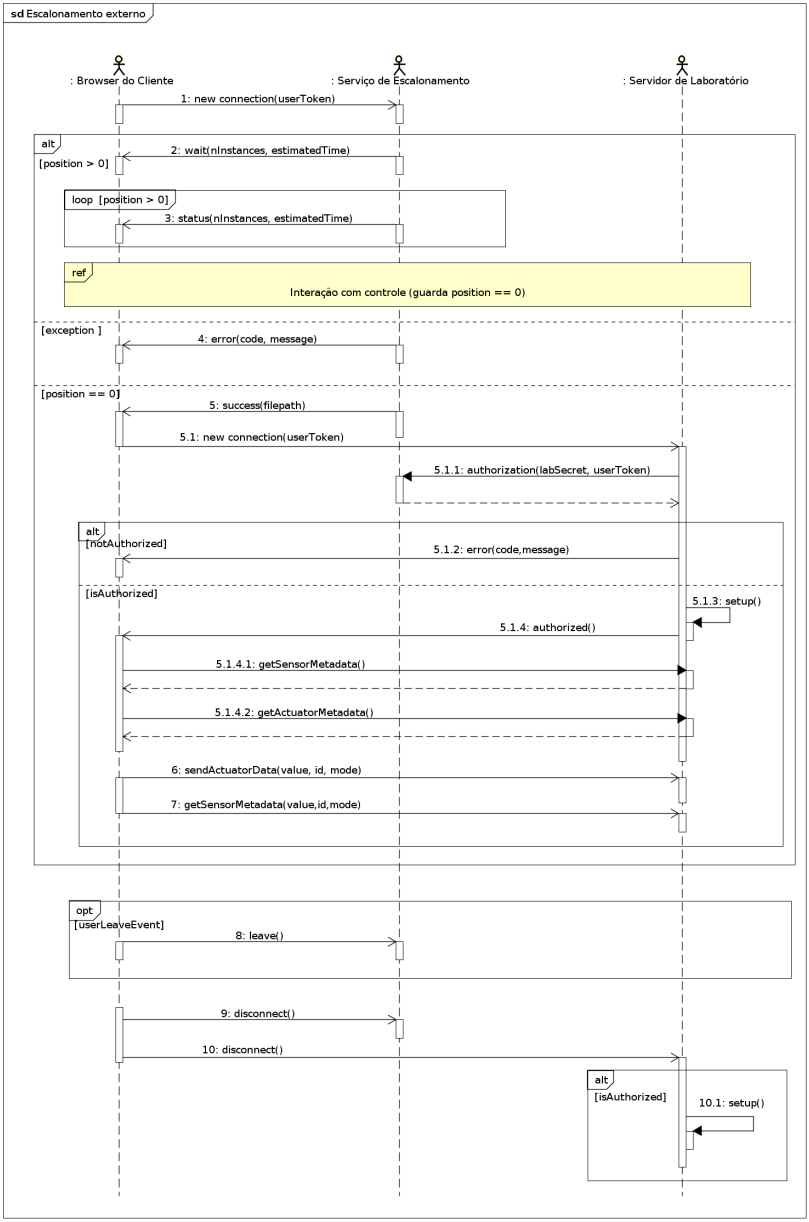

Figura 11 – Diagrama de sequência do serviço escalonamento de usuário externo ao dispositivo inteligente. Fonte: Autor.

O mesmo conceito de balanceamento de carga foi utilizado para distribuir a fila entre laboratórios semelhantes. No entanto, optou-se em usar uma única fila para cada grupo de laboratórios semelhantes em vez de direcionar para filas distintas, que seriam de responsabilidade de cada servidor de laboratório, como uma aplicação de balanceamento de carga faria.

A simples distribuição de carga por fatores como número de usuários conectados ou tempo de resposta seria indesejável, pois pode gerar situações em que um laboratório pode ocioso enquanto outro está sobrecarregado devido ao comportamento aleatório dos usuários. Assim, a coordenação dos usuários conectados e dos laboratórios disponíveis fica sob responsabilidade de uma aplicação auxiliar e externa ao servidor de laboratório, que direciona o usuário ao laboratório disponível.

Essa abordagem exige a sincronização entre três aplicações: cliente, servidor de escalonamento e servidor de laboratório. O diagrama de sequência na figura 11 apresenta uma visão geral das mensagens trocadas. Antes da interação apresentada no diagrama, cada usuário do RELLE recebe um *token* de sessão que é utilizado para registrar-se na fila de espera, isto é, a primeira mensagem oriunda do cliente para o serviço de escalonamento. O cliente mantém um canal WebSocket aberto com o serviço de escalonamento e, caso haja queda de conexão, ele tentará reestabelecer sua posição na fila ou como controlador desde que a reconexão seja realizada antes de expirar o *timeout* da conexão anterior, como mostrado no diagrama da figura 12.

Quando o usuário obtém permissão para controle do laboratório por um determinado bloco de tempo, o mesmo token é utilizado para acesso ao laboratório. No servidor de laboratorio há a necessidade de um autorizador, que consulta o serviço de escalonamento com o token do usuário e recebe de volta a sinalização da autorização e duração da experiência. A mensagem de autorização enviada usando o protocolo HTTP e, além do token do usuário, deve conter uma chave para autenticação do laboratório que foi previamente armazenada no banco de dados do serviço de escalonamento.

Em termos de projeto são separados dados comuns às instâncias e comum ao laboratório, como ilustra o diagrama entidade-relacionamento da figura 13. Cada tipo de laboratório é identificado por um *namespace* que é usado na URL pelo cliente, a fila para o experimento Painel CC, por exemplo, é acessada no caminho *ws://relle.ufsc.br/queue/ID*, sendo *ID* o número identificador deste experimento. A aplicação está estruturada em quatro classes principais, como apresentada na figura 14, e realiza o escalonamento usando canais WebSockets com clientes. A aplicação foi escrita Javascript para ser executada no ambiente de execução Node.js e usa um banco de dados SQL[7].

[7] `https://gitlab.com/joaopaulocl/ssi`

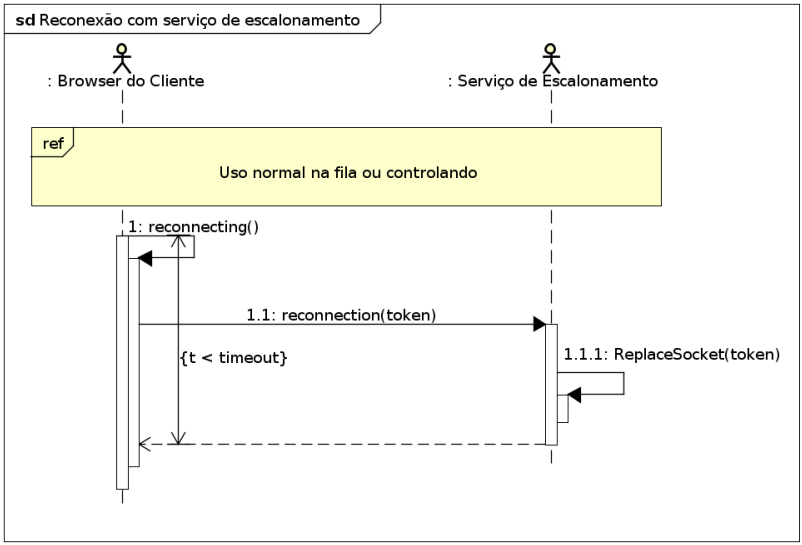

Figura 12 – Diagrama de sequência para reconexão do cliente no serviço de escalonamento. Fonte: Autor.

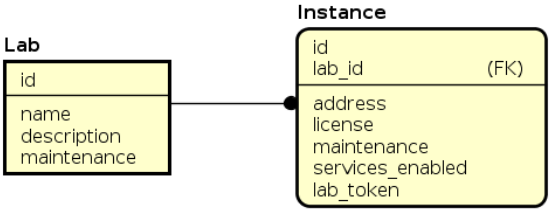

Figura 13 – Diagrama de entidade-relacionamento de instâncias de experimentos. Fonte: Autor.

3.1.2.2 Interno ao servidor de laboratório

Embora a abordagem anterior mostre-se como uma solução para escalonamento de múltiplas instâncias e para redução de requisições no servidor de laboratório, há casos em que a independência de serviços externos é mais relevante para o cumprimento dos requisitos do sistema. Esta situação ocorre

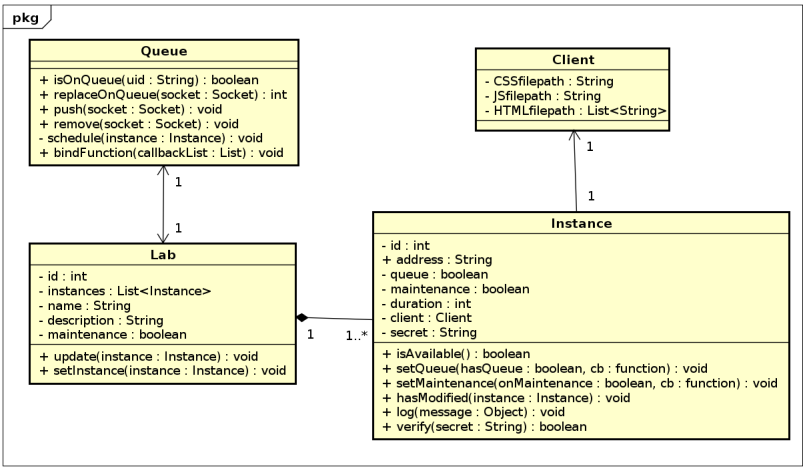

Figura 14 – Diagrama de classes da aplicação de escalonamento externo. Fonte: Autor.

no laboratório remoto de Arduino que foi desenvolvido no RExLab[8]. No projeto desse laboratório foram desenvolvidos uma aplicação cliente chamada block.ino e um servidor de laboratório para serem utilizados como um sistema autônomo. Ambas aplicações têm código-fonte aberto e podem ser implantadas e executadas sem necessidade de serviços externos, dependências externas ou mesmo de acesso à Internet.

De maneira geral, o laboratório de Arduino é composto por um kit com placa Arduino, sensores, atuadores e outros componentes eletrônicos. O servidor de laboratório, neste caso, atua no gerenciamento das ferramentas de desenvolvimento e programação do microcontrolador, além de ser a ponte para troca de informações entre o Arduino e o usuário no momento de execução. Os usuários podem compilar seus códigos em qualquer momento, mas somente podem enviá-lo para o microcontrolador e testar seus programas dentro de uma sessão , este tempo pode ser extendido caso não haja outros usuários requisitando acesso no momento.

A segunda abordagem traz o escalonamento para a mesma aplicação que gerencia o controle do laboratório, exigindo apenas sincronização entre a aplicação cliente e o servidor de laboratório. A formalização da interação entre as duas aplicações é apresentada no diagrama de sequência (figura 15). O usuário do aplicativo recebe um *token* de sessão que é utilizado para registrar-

[8] `https:gitlab.com/joaopaulocl/blockino-backend`

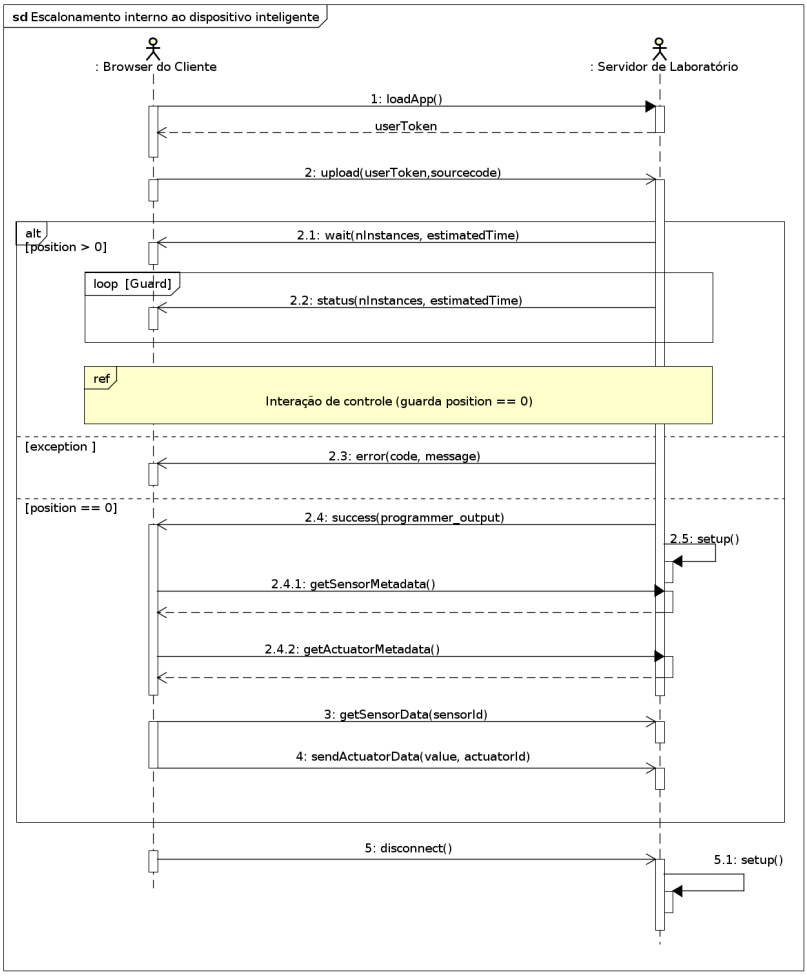

Figura 15 – Diagrama de sequência do serviço de escalonamento de usuário interno ao dispositivo inteligente. Fonte: Autor.

se na fila apenas no processo de envio do código, substituindo a primeira mensagem para o serviço de escalonamento.

Apesar de ambas abordagens usarem o mesmo módulo, suas funções podem ser extendidas, assim como a identificação de cada mensagem pode ser customizada. Por exemplo, o registro na fila, que na abordagem da seção anterior chamava-se *new connection* e retornava dados sobre a aplicação cli-

ente na mensagem *success* quando o usuário obtinha acesso, foi substituída pela mensagem *upload*, que envia o código para a placa através de funções vinculadas à sessão do usuário após o evento de escalonamento e retorna na mensagem *success* com informações sobre o processo de programação na placa Arduino.

Como todas mensagens são identificadas por um nome, o cliente mantém somente um canal WebSocket aberto com o servidor de laboratório e usa-o tanto para funções do laboratório quanto para funções de gerência do usuário. Assim como ocorre com o serviço de escolanamento externo, a aplicação cliente tentará reestabelecer sua posição na fila se houver queda de conexão do usuário, como mostrado anteriormente na figura 12.

3.1.2.3 Testes e comparação

O novo sistema de escalonamento traz melhorias em relação à usabilidade por tratar eventos de conexão, notificações e desconexão dos usuários muito mais rapidamente do que a aplicação anterior implementada sobre HTTP. A redução do tempo para reconhecimento desses eventos tem efeito sobre o tempo em que o experimento fica ocioso entre a desconexão de um usuário e a conexão do próximo usuário esperando na fila.

Foram realizados dois testes a fim de avaliar o comportamento dos serviços de escalonamento. O primeiro tem o objetivo de comparar a antiga implementação de escalonamento com a atual e foram medidos tempo entre a desconexão de um cliente e a conexão de um novo. O segundo teste avalia o tempo de resposta nas duas abordagens de escalonamento, sendo uma implantada em servidor com alta capacidade e outra embutida no servidor de laboratório e implantada em Raspberry Pi.

Para execução de ambos os testes foi utilizada uma máquina com processador Intel Core i5-3317U CPU@1.70GHzx4 e 8GB de memória RAM executando Ubuntu 14.04 LTS 64 bits e Mozilla Firefox 47.0, sendo conectada em uma rede diferente da rede em que os servidores estão conectados. Cada valor obtido refere-se à média de 10 amostras para reduzir a tendência estatística.

Em testes realizados sobre o antigo sistema de escalonamento era percebido um tempo médio de 5.46 segundos (LIMA et al., 2015). O mesmo teste realizado sobre o novo sistemas apresenta tempo médio de 0.8 segundos. Esta melhoria deve-se ao uso do protocolo WebSocket, que adiciona características de tempo real web à aplicação cliente e servidor e afetam positivamente na experiência de usuário.

No segundo teste foi obtido o tempo médio entre mensagem de en-

trada na fila e retorno de mensagem de espera ou acesso. Foram disparadas requisições concorrente, variando entre 10 e 100 clientes, aumentando com um passo de 10. Os testes foram realizados sem verificação de token no RELLE, pois o objetivo é analisar a aplicação escrita em Node.js sem causar sobrecarga para o servidor Apache executado na mesma máquina (infraestrutura e descrição da implantação do sistema em produção são apresentados na seção 3.3).

Na figura 16 são apresentadas duas curvas: *Servidor Comum*[9] e *Raspberry Pi*[10] que apresentam respectivamente o tempo de resposta para o serviço de escalonamento externo e interno ao servidor de laboratório para execução do mesmo serviço. Durante a execução do teste foram observados limites de conexões WebSocket identificados como *falha de conexão WebSocket: WebSocket foi fechado antes de conexão ser estabelecida*.

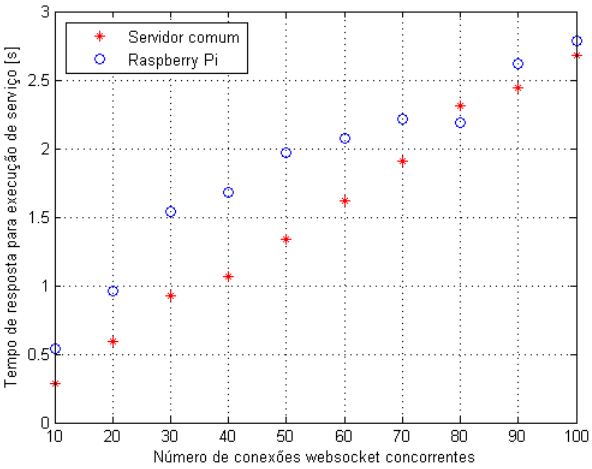

Figura 16 – Tempo de resposta em relação ao número de clientes concorrentes para duas implantações de escalonamento. Fonte: Autor.

Este erro de conexão deve-se à grande quantidade de conexões simultâneas, pois a criação de um canal WebSocket com Socket.io inicia cinco requisições HTTP, sendo que uma delas é atualizada para WebSocket e torna-se o canal WebSocket propriamente dito. O restante das requisições HTTP

[9] O computador utilizado no teste possui processador Intel Xeon E5-2609 v3 de 1.9 GHz e cache de 15 MB, memória RAM de 8GB e executa Ubuntu Server 12.04.5 LTS e Node.js v.0.12.9

[10] O computador utilizado é um Raspberry Pi 2 modelo B executando Raspbian 4.1.17-v7+ e Node.js v.0.12.9

fazem parte da biblioteca Socket.io e tem a função de manter compatibilidade com navegadores que não suportam WebSockets. Além disso, o mesmo teste com intervalo de 500 ms entre cada nova conexão não apresentaram conexões recusadas. Outros testes com variação do intervalo de tempo entre requisições e número de clientes conectados podem ser realizados identificar a carga máxima ideal para essa aplicação sem ocorrer recusa de conexão.

Além disso, o número de conexões recusadas pode estar relacionado com a capacidade de memória da máquina e configurações do sistema operacional do servidor, visto que as porcentagens de conexões recusadas são maiores para a fila implantada no Raspberry Pi do que em Servidor Comum, como apresentado na figura 17. No entanto, os relatos encontrados na página de rastreamento de erros do Socket.io e em fóruns de desenvolvimento não identificam a origem deste erro.

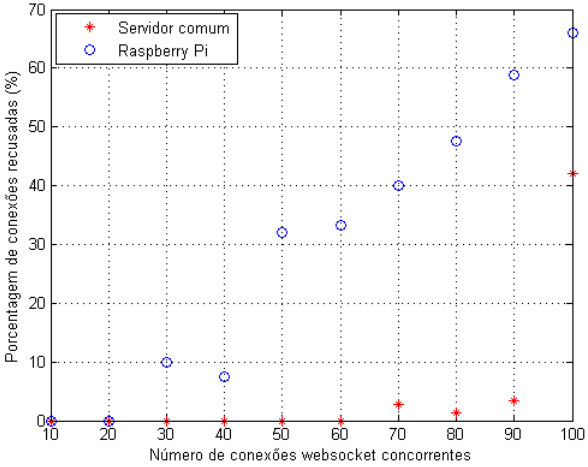

Figura 17 – Porcentagem de conexões recusadas em relação ao número de clientes concorrentes. Fonte: Autor.

Ambas abordagens obtiveram tempos médios de resposta satifatórios e linearmente dependente do número de clientes conectados. A diferença entre o tempo de resposta entre as duas abordagens é desconsiderável para usá-lo como parâmetro de recomendação entre uma ou outra abordagem. Porém, as porcentagens de conexões recusadas indicam que a aplicação com escalonamento interno estará sobrecarregada mais rapidamente.

Além disso, esses testes não avaliaram o tempo de resposta para todo cenário, em que o tempo de resposta de requisições de controle do labo-

ratório também podem ser medidos. Neste caso, a aplicação de laboratório com escalonamento interno terá sobrecarga adicional para requisições como a recuperação de dados de sensores.

A abordagem de escalonamento externo, além de possibilitar o balanceamento de carga entre instâncias, também evita sobrecarga do servidor de laboratório para gerenciar fila. No entanto, esse modelo exige que o servidor de laboratório seja coordenado somente por um servidor de escalonamento. Assim, em termos de escalabidade para toda arquitetura do RELLE, o escalonamento externo justifica-se como melhor opção.

3.2 SERVIDOR DE LABORATÓRIO REMOTO

A partir de características específicas de cada experimento, diferentes sensores, atuadores e rotinas de controle são usadas para mensurar nível de tensão com voltímetro, posicionar um servomotor ou ligar ou desligar um aquecedor, por exemplo. No entanto, para habilitar o acesso desses recursos via Internet, um servidor deve oferecer uma abstração dos sensores e atuadores de cada laboratório aos usuários. Assim, um dos componentes da infraestrutura dos laboratórios remotos do GT-MRE é um servidor que provê um conjunto de operações e gerencia um recurso físico.

Cada laboratório remoto foi projetado com base com em Softwares abertos e protocolos recomendados pela *World Wide Web Consortium* a fim de facilitar a replicação e integração destes laboratórios em ambientes de aprendizagem distribuídos para ensino e aprendizagem. A seguir serão descritos os desenvolvimentos realizados ao longo da primeira e segunda fase do GT-MRE e, por fim, a adaptação dos servidores de laboratório para o paradigma de dispositivos inteligentes.

3.2.1 Tecnologias utilizadas

A escolha das ferramentas de desenvolvimento e produção considera quatro principais requisitos do projeto: (a) desempenho satisfatório em SBC ou computadores mais velhos, (b) baixa curva de aprendizagem, (c) suporte ao hardware do Raspberry Pi e (d) suporte aos protocolo de transferência de dados exigidos pelo paradigma de dispositivos inteligentes.

O primeiro requisito deve-se ao baixo custo e alta disponibilidade da SBC Raspberry Pi para ser usado como um servidor web leve e autônomo para cada experimento. Considerando a especificação de dispositivos inteligentes, os protocolos de transferência de dados (HTTP e WebSocket) e for-

mato de dados em JSON devem ser suportados.

Visto que servidores como o servidor HTTP Apache e Tomcat são *multi-threaded* e requerem capacidade de memória e processamento consideravelmente alta, foram analisadas algumas tecnologias adequadas para computadores de baixo poder de processamento. Baseando-se em testes disponíveis em relatório técnicos (TILKOV; VINOSKI, 2010; LEI; MA; TAN, 2014), foi escolhido o Node.js como framework para construção dos servidores. Além disso, foram considerados os pontos levantados pelos trabalhos de Bosák e Žáková (2015), Wang et al. (2015) quanto à adequação do Node.js para sistemas de laboratórios remotos.

O Node.js é um ambiente de execução de plataforma cruzada e código aberto para construir aplicações de rede e servidores escaláveis. Ele possui diversas bibliotecas embutidas para funções de entrada e saída em arquivos, protocolos de rede, fluxo de dados e criptografia, por exemplo, além de um repositório público específico para Node.js.

Um dos recursos importantes para acelerar o processo de desenvolvimento de servidores de laboratório são os complementos nativos. Eles possibilitam o reuso de bibliotecas escritas em C/C++ e garantem performance de algoritmos escritos nessa linguagem. Outra característica importante é o modelo orientado a eventos do Node.js e suporte a operações de entrada e saída não bloqueantes para otimizar a taxa de transferência e escalabildade da aplicação. O WebSocket e o paradigma orientado a eventos juntos são apropriados para aplicações de laboratórios remotos porque estes sistemas geralmente precisam entregar seus resultados ao cliente quando um evento é disparado, como a detecção de movimento ou campo magnético.

A implementação de APIs HTTP e WebSocket pode ser feita apenas com módulos providos pelo ambiente de execução, porém o repositório tem bibliotecas de terceiros que facilitam o desenvolvimento, como Express[11] e socket.io. A biblioteca socket.io é composta por dois componentes, biblioteca cliente e servidor, e são usadas para construir aplicação web para laboratórios remotos em tempo real usando WebSocket, e também *polling* HTTP como compatibilidade reversa.

3.2.2 Implementação base

A aplicação de servidor contém três camadas de software diferentes e independentes como mostrada na figura 18: (a) camada de controle e aquisição, (b) de lógica de acesso e (c) de rede. A primeira camada consiste

[11] Express é um framework MVC de aplicação web. Disponível em: `http://expressjs.com/pt-br/`

em uma API para comunicação com hardware capaz de operar sensores e atuadores, que será apresentada na seção 3.2.3. A camada (c) é responsável pela exposição das operações sobre o equipamento físico usando WebSocket. A camada (b) provê funções específicas de cada laboratório e recursos que podem ser reutilizados. Todas camadas seguem o paradigma de orientação a eventos.

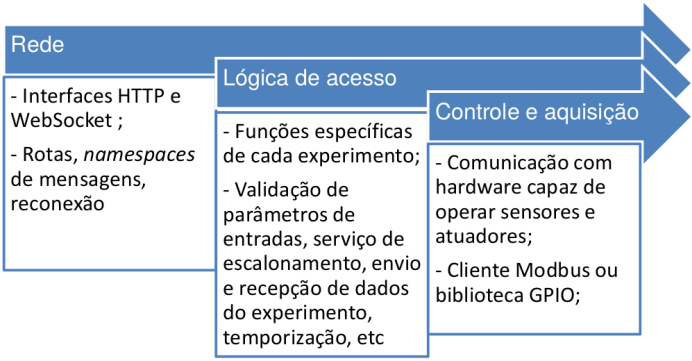

Figura 18 – Camadas de implementação da aplicação de laboratório remoto. Fonte: Autor.

Foram utilizados o framework Express e a biblioteca socket.io na camada (c), onde são definidos parâmetros para o protocolo HTTP e WebSocket, como rotas, *namespace* de mensagens, intervalo de mensagens *heartbeat* e *timeout* de conexão. Antes da adaptação das aplicações para o paradigma de dispositivos inteligentes já havia sido adotado o WebSocket e JSON como padrões obrigatórios para os laboratórios do GT-MRE.

Na camada (b) são implementadas funções específicas de cada experimento, como a validação de parâmetros de entrada, agendamento e temporização de acesso aos sensores, envio de sinais para atuadores, rastreamento de estado do experimento e tratamento de exceções de hardware. Nesta camada também é implementado o controle de concorrência ao recurso conforme a abordagem escolhida, isto é, escalonamento interno ao servidor de laboratório ou externo. Como todos laboratórios seguem o mesmo modelo de acesso, é possível reutilizar código para esta funcionalidade.

A implementação do escalonamento externo é baseada na autenticação por token provido pelo RELLE e validado pelo servidor de laboratório, como discutido na seção 3.1.2.1, e são exemplificados pelos experimentos de física e biologia. Já o controle de acesso no próprio dispositivo inteligentes, como explicado na seção 3.1.2.2, é exemplificado pela implementação do Labo-

ratório de Desenvolvimento em Arduino, pois neste encontra-se um modelo de acesso diferente dos anteriores.

A autorização de sessão no servidor de laboratório garante a integridade do acesso exclusivo, já que o dispositivo exposto como um serviço pode ser utilizado concorrentemente por outro cliente. Apesar de algumas funcionalidades poderem ser utilizadas no modo observador, como consultar o estado das chaves e metadados, as funcionalidades de controle necessitam de consulta ao sistema de fila.

Além disso, o Raspberry Pi também realiza a captura de vídeo de uma câmera USB genérica usando o software Motion[12] e provê a transmissão de vídeo no formato MJPEG. Este servidor de streaming é independente da aplicação de controle, mas, se habilitado, o Motion fornece uma API HTTP para ajustar parâmetros como taxa de quadros, brilho e qualidade de compressão, por exemplo. Como alguns navegadores não suportam transmissão MJPEG também são fornecidas imagens atualizadas a uma taxa customizável no servidor para que a aplicação cliente atualize as imagens em um intervalo adequado.

3.2.3 Interface com hardware de experimentos físicos

A camada de comunicação com o hardware pode agir diretamente sobre os pinos da placa que roda a aplicação ou em outra plataforma de desenvolvimento que podem ser capazes de operar sensores e atuadores. Apesar de ter sido seguida uma abordagem modular para construção, o acesso ao hardware pode variar para cada experimento. Para laboratórios simples o acesso pode ser através de pinos na placa do próprio Raspberry Pi, como transdutores que usam GPIO ou portas para comunicação. Outra possibibilidade é prover uma interface para os módulos de entrada e saída via USB ou UART usando uma placa Arduino ou outra placa customizada para controle e aquisição, por exemplo.

Os dois casos ocorrem nos laboratórios do GT-MRE. As bibliotecas para Modbus[13] e GPIO[14], ambas em C++, são usadas na implementação dos laboratórios de física e química. Já o laboratório de Arduino usa a biblioteca *serialport*[15] e o software *avr-dude* do pacote de ferramentas da AVR. A orientação a eventos também foi utilizada, aproveitando interrupções de hardware e software para entregar mensagens em tempo real web.

[12] `http://www.lavrsen.dk/foswiki/bin/view/Motion/WebHome`
[13] `https://gitlab.com/joaopaulocl/modbus-client`
[14] `https://gitlab.com/joaopaulocl/RaspberryPi-GPIO`
[15] `https://github.com/EmergingTechnologyAdvisors/node-serialport`

O código-fonte desenvolvido para gerência dos sensores e atuadores são complementos para o NodeJS escritos em C++, também chamados de objetos compartilhados de vínculo dinâmico. Apesar de não seguir criteriosamente o conceito de orientação a objeto, a abstração de cada experimento físico é representada por um objeto com métodos e atributos intrínsecos de cada um. Por exemplo, métodos de *get* e *set* para saídas digitais, *get* para valores de sensores.

3.2.4 Laboratórios desenvolvidos

Os experimentos criados do GT-MRE baseiam-se em recomendações de docentes e em experimentos anteriores ao GT. Em cada experimento, uma placa de controle e aquisição (PAC) é responsável por receber sinais dos sensores e transmitir à aplicação hospedada no Raspberry Pi que gerencia o experimento. Os sensores e atuadores também foram construídos pelo GT-MRE para adequar as necessidades de cada experimento.

O esquema dessas placas e a função por experimento são descritos na documentação do manual técnico[16], assim como listas de sensores e atuadores, interface de comunicação e, se houver PAC, protocolo e descrição de retorno de funções.

Em um primeiro momento, para cada experimento foi desenvolvida uma aplicação seguindo um template em Node.js, que serviu como ponto de partida para cada novo experimento. A implementação de cada aplicação está disponível em repositório público[17].

3.2.5 Adaptação para Dispositivos Inteligentes

Como apresentado na seção 1.3, um dos objetivos deste trabalho é reprojetar as aplicações dos laboratórios do GT-MRE com a inclusão da definição de serviço, de metadados e de protocolos conforme a especificação de dispositivos inteligentes. Do conjunto de laboratórios disponíveis foi selecionado o Painel CC por conter a PAC, dois tipos de sensores e um tipo de atuador, além de seus componentes serem bastante estáveis e representativos em relação aos outros laboratórios.

Primeiramente foram realizadas modificações na infraestrutura de rede

[16] Os manuais constam na página de acesso de cada experimento na aba **Documentação**. Por exemplo, o Manual Técnico do Painel CC está disponível em `http://relle.ufsc.br/labs/1>`

[17] `https://github.com/RExLab`

Tabela 3 – Experimentos remotos desenvolvidos pelo GT-MRE e descrição de seus componentes. Fonte: Autor.

Experimento	**Interface/Protocolo**	**Componentes**
Banco Óptico	UART / Modbus	PAC, módulos de relés, sensor de posição
Condução de calor	UART / Modbus	Módulo de relés, PAC, visores e termômetros
Disco de Newton	GPIO	Motor de corrente contínua e *driver*
Fotovoltaico	UART / Modbus	PAC, módulos de relés, servomotor, multímetro, painel de LEDs e visores
Microscópio Remoto	UART / Modbus	PAC, servomotor e sensor de posição
Laboratório de Arduino	USB	Placa Arduino que pode conter diferentes combinações de sensores, atuadores e componentes eletrônicos
Painel Elétrico CA	GPIO	Módulos de relés
Painel Elétrico CC	UART / Modbus	Módulos de relés eletromecânicos, PAC, multímetros e visores
Plano Inclinado	UART / Modbus	PAC, servomotores, sensor de peso, sensores ópticos, acelerômetro e visores
Propagação de calor	UART / Modbus	Módulo de relés, PAC, visores termômetros

e atribuídos IP público e domínio DNS para cada Raspberry Pi. Esta alteração é indicada como boa prática na especificação pois confere identidade única a cada laboratório. Em seguida foram levantadas e avaliadas implementações de dispositivos inteligentes disponíveis em repositório de código-fonte aberto[18]. As implementações foram feitas para diferentes plataformas, como Desktop, myRIO NI, BeagleBone e Raspberry Pi, e em duas linguagens, LabVIEW e Node.js.

Para adequação dos laboratórios do GT-MRE para o paradigma de dispositivos inteligentes foram utilizados a especificação original, exemplos

[18] `https://github.com/go-lab/smart-device/`

do projeto Go-Lab[19] e um exemplo disponibilizado pelo laboratório UNEDLabs[20]. Foi possível reutilizar totalmente a lógica para validação de parâmetros de entrada, comunicação com a PAC e eventos disparados com a chegada de mensagens da PAC. Os serviços opcionais, como o serviço para recuperação de clientes e de modelos matemáticos e gráficos, foram omitidos. Apenas os serviços obrigatórios foram implementados para fornecer as mesmas funções da implementação do servidor de laboratório anterior. O código-fonte foi disponilizado em repositório público[21].

Tabela 4 – Serviços do laboratório Painel CC compatíveis com a especificação de dispositivos inteligentes. Fonte: Autor.

Serviço	**Descrição**
getSensorMetadata	Retorna lista de sensores (voltímetros e amperímetros) com descrição, nome, identificador, modo de acesso e estrutura de dados para valores, que inclui o grandeza física, *timestamp* da última medida e faixa de valores, por exemplo
getActuatorMetadata	Retorna lista de atuadores (chaves do tipo relé) com descrição, nome, identificador, modo de acesso e estrutura de dados para valores, que inclui grandeza física e faixa de valores permitidos.
getSensorData	Retorna array de valores de multímetros
sendActuatorData	Envia dados para um ou mais atuadores

A aplicação cliente do Painel CC teve modificações para suportar dispositivos inteligentes e usa a biblioteca *Smart Device JS*[22] para reuso de componentes de comunicação para qualquer dispositivo compatível com a especificação. Essa biblioteca é uma versão da biblioteca *swagger-js*, porém com suporte para WebSocket. O cliente utiliza um arquivo de descrição de serviço descrito conforme a especificação de metadados de dispositivo inteligente baseada em Swagger.

Apesar de a migração ter ocorrido apenas para o experimento Painel Elétrico CC, já pode-se antecipar algumas dificuldades. O paradigma con-

[19] O projeto Go-Lab disponibiliza a especificação de metadados neste repositório do GitHub: `https://github.com/go-lab/smart-device-metadata`. Exemplos para Node.js e plataforma myRIO são disponibilizados neste repositório: `https://github.com/go-lab/smart-device`

[20] A implementação em Node.js e cliente web de dispositivos inteligentes do UNEDLabs da Universidad Nacional de Educación a Distancia tem seu código-fonte em repositório público do GitHub: `https://github.com/UNEDLabs/smartdevice`

[21] `https://gitlab.com/joaopaulocl/smartdevice-template`

[22] `https://github.com/UNEDLabs/smartdevice/tree/master/smartdevice-js`

sidera que um laboratório remoto é composto sumariamente por sensores e atuadores, e seus clientes devem ter a visão em termos desses componentes independente do projeto de hardware e funcionalidades implementadas por software específicas de cada laboratório. No entanto, a especificação dos serviços *getSensor** e *getActuator** não é suficiente para alguns laboratórios.

O servidor do Laboratório de Arduino, por exemplo, possui funções que devem ser oferecidas como serviço, como compilar código-fonte e programar o microcontrolador e transferir dados através de uma interface de comunicação, porém não se caracterizam como sensor ou atuador. A mesma situação possivelmente ocorreria em outros experimentos em que o objetivo é programar ou sintetizar o comportamento de um controlador, ou seja, suas funções não são pré-definidas, e o usuário define se haverá troca de mensagens em tempo de execução e como elas são estruturadas.

3.2.6 Testes e avaliação

O principal objetivo dos testes de conexão foi avaliar o comportamento da aplicação de servidor para números crescentes de requisições. Como as operações desta aplicação são protegidas por controle de concorrência, será avaliado o tempo de resposta de operações oriundas de um cliente em diferentes frequências. Foram realizados três testes com um elevado número de requisições que solicitam o serviço de um único servidor de laboratório.

O número de requisições variou entre 50 e 400, aumentando com um passo de 50. O cliente foi conectado em uma rede diferente da rede em que os laboratórios estão conectados. A fim de reduzir a tendência estatística, cada valor obtido refere-se ao resultado da média de 10 execuções. O primeiro teste (Teste 1) avalia o tempo de resposta de requisição de um valor de um sensor por parte do cliente, o segundo (Teste 2) identifica o tempo de resposta de requisições partindo da aplicação de servidor e o sistema embarcado do experimento e a última avaliação (Teste 3) avalia somente o tempo de resposta da aplicação sem retorno de dados (*round trip*).

Para execução dos Testes 1 e 3 foram criados scripts em Javascript como cliente e executados em navegador Mozilla Firefox com requisições para a aplicação de um servidor de experimento. No teste 2 foram realizadas requisições somente a partir da aplicação de servidor de experimento e hospedada em um Raspberry Pi 1 modelo B+ executando Raspbian 4.1.19 e Node.js v0.12.9. A aplicação de servidor faz requisições usando protocolo Modbus serial para placa de aquisição e controle que contém um multímetro conectado a ela. A aplicação cliente foi executada em máquina com processador Intel Core i5-3317U CPU@1.70GHzx4 e 8GB de memória RAM executando

Ubuntu 14.04 LTS 64 bits e Mozilla Firefox 47.0.

Nos Testes 1 e 3, o objetivo foi medir o custo de requisições através da Internet com e sem aquisição de dados, respectivamente. O teste 3 também serve como comparativo para latência da rede, o que poderá variar dependendo de característica da rede em que os usuários estão conectados. Já o teste 2 apresenta o custo de requisições para aquisição de dados local usando protocolo serial, ou seja, inclui tempo para aquisição de dados, processamento e transferência de valores através de uma interface USART com *baudrate* de 57600[23].

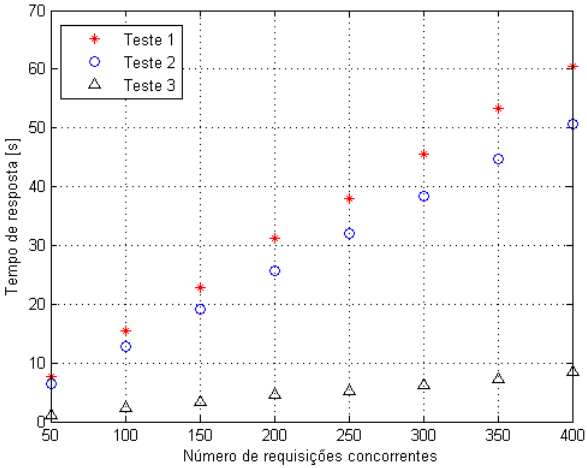

Figura 19 – Tempo de resposta em relação ao número de requisições simultâneas para o Teste 1, 2 e 3. Fonte: Autor.

A partir destas repetições, o Teste 1, 2 e 3 apresentaram tempo médio para uma requisição de 153ms, 128ms e 22ms, respectivamente. O comportamento do sistema com o acréscimo de carga é praticamente linear para requisições de dados de sensores, como mostra a figura 19, pois a aquisição é uma função bloqueante. Os tempos de respostas são bastante satisfatórios e são sempre dependentes do modo e tempo de acesso à placa de aquisição e controle, isto é, se é bloqueante ou não.

A partir do tempo médio de uma requisição é possível inferir a frequência máxima recomendada para atualização de sensores e atuadores. No experi-

[23]Esta taxa foi escolhida por ser o máximo valor suportado na configuração padrão do Raspberry Pi. Se alterado no arquivo de configuração de inicialização do sistema operacional, podem ser atingidas taxas mais altas.

mento avaliado, por exemplo, a frequência máxima apresentada para o usuário deve ser de 7 requisições por segundo. Os limites de frequência de requisições e tempo de resposta podem ajudar a orientar melhor o projeto das aplicações de servidor de laboratório, bem como dimensionar o número de recursos para escalar o sistema.

Esse teste apenas identificou o comportamento da aplicação de laboratório remoto para um cliente realizando requisições sob diferentes frequências, visto que é esperado que cada usuário tenha acesso exclusivo ao equipamento. Além disso, podem ser realizados testes similares para avaliar o tempo de resposta para o cenário de colaboração entre usuários. Assim, em vez de as requisições serem originadas apenas de um canal WebSocket, serão vários canais requisitando dados do laboratório remoto.

3.3 INFRAESTRUTURA E IMPLANTAÇÃO

O ambiente de aprendizagem RELLE hospeda as aplicações cliente para acesso aos laboratórios remotos, como mencionado anteriormente. O diagrama de componentes apresentado na figura 9 representa todos os componentes de hardware e de software envolvidos na arquitetura de experimentação remota do GT-MRE.

Cada aplicação cliente é composta por um conjunto de arquivos HTML, CSS e Javascript e comunica-se com o servidor de laboratório através da *API WebSocket*, que segue a especificação de dispositivos inteligentes, e com o serviço de escalonamento através de uma interface WebSocket chamada *API FCFS*.

O serviço de escalonamento atende às requisições de clientes autorizados após validar o identificador de sessão com a aplicação RELLE através da interface *Validação de Sessão*. Assim, o serviço de escalonamento obtém informações do usuário e pode servir como autorizador para os servidores de cada laboratório através da interface *Autorização de acesso*.

Cada laboratório, além de prover *API WebSocket*, pode opcionalmente transmitir imagens do equipamento, interface chamada de *MJPEG Streaming*, ao cliente. As interfaces de comunicação entre o servidor e o hardware do laboratório podem mudar conforme as necessidades de cada laboratório, porém são utilizadas uma destas duas opções: *API Modbus* ou *API para hardware do Raspberry Pi*.

Os diagramas de implantação do RELLE e serviço de escalonamento (figura 20) e de servidores de laboratório (figura 21) apresentam os componentes internos de suas aplicações. As dependências externas referem-se a códigos que são reutilizados, todos disponíveis em repositório aberto. Na

figura 20 são apresentadas as interfaces entre as aplicações e o banco de dados, assim como os modelos relacionais utilizados. Na figura 21 é apresentada a organização da aplicação genérica e as duas opções de interface de comunicação com hardware do laboratório.

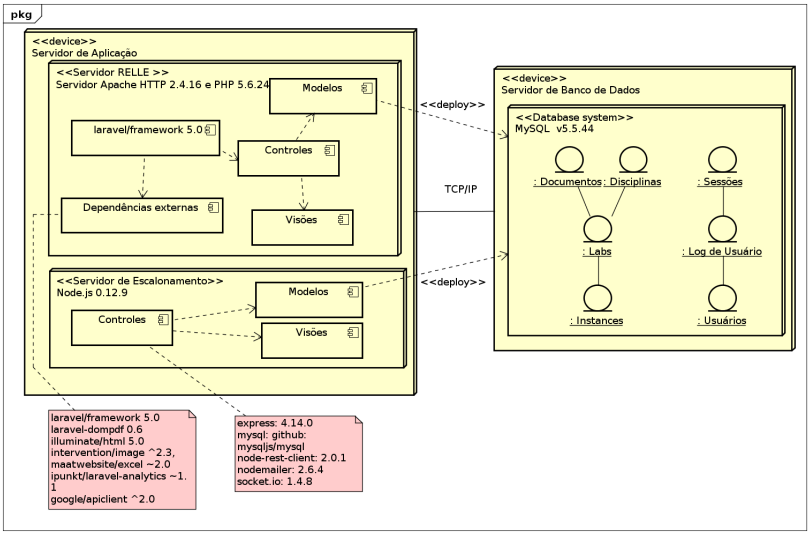

Figura 20 – Diagrama de implantação do RELLE. Fonte: Autor.

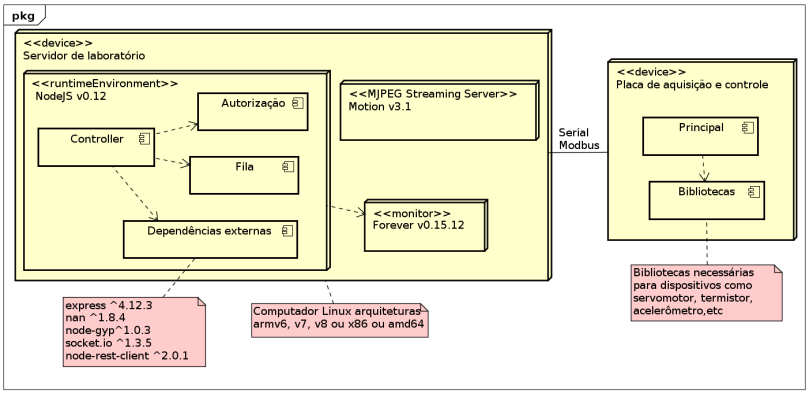

Figura 21 – Diagrama de implantação de um servidor de laboratório. Fonte: Autor.

4 CONSIDERAÇÕES FINAIS

A seguir são apresentadas as contribuições deste trabalho e propostas para melhorias e trabalhos futuros.

4.1 CONTRIBUIÇÕES DESTE TRABALHO

Considerando as dificuldades encontradas no projeto de laboratórios remotos devido à falta de padronização e de estratégias de projeto de software, este trabalho teve como finalidade projetar uma aplicação modelo para o desenvolvimento de laboratórios remotos e o compartilhamento em ambientes de aprendizagem.

Para isto, foram destacados como objetivos específicos:

(a) Identificar questões relacionadas ao projeto de experimentos remotos com capacidade para serem compartilhados em diferentes LMSs: ao qual analisou e identificou na literatura os modelos de referência, como os principais RLMS existentes, o modelo LaaS e o paradigma de dispositivos inteligentes.

(b) Detalhar os requisitos e arquitetura da plataforma de experimentação do GT-MRE: apresentou a arquitetura da solução, compreendendo o ambiente de aprendizagem RELLE e servidor de laboratório, tecnologias utilizadas, aspectos de implementação e implantação.

(c) Projetar sistema de escalonamento compatível com requisitos do projeto GT-MRE: foram descritas duas abordagens de escalonamento de usuários, bem como foram mostrados testes e comparações sobre ambas abordagens e também sobre a implementação atual em relação à versão anterior.

(d) Projetar uma aplicação de servidor de laboratório baseada no paradigma de dispositivos inteligentes: foi relatado o uso do paradigma de dispositivos inteligentes em um experimento, o Painel Elétrico CC, e analisadas as contribuições deste paradigma em aspectos de reusabilidade de código cliente, simplicidade e padronização de interação com serviços.

Portanto, a partir de descrições e testes apresentados, este trabalho colaborou com a padronização e reuso de software em vários níveis para os laboratórios do GT-MRE e seu sistema de acesso.

4.2 OPORTUNIDADES DE TRABALHOS FUTUROS

Ao longo do texto são evidenciadas algumas limitações aqui resumidas:

- Visto que usuários convidados não possuem identificação e não foram desenvolvidas questões de segurança para laboratórios públicos, é possível encontrar vulnerabilidade e forte dependência entre os componentes no método de autorização atual.
- Apesar de parte da comunicação entre o cliente e o servidor de laboratório ser automatizada por uma biblioteca e um arquivo de descrição de serviço, os metadados no lado do servidor ainda são fixos e incorporados no código-fonte da aplicação.
- O processo de criação de interface de usuário é dispendioso e inflexível para atender usuários de diferentes níveis ou sob abordagens pedagógicas diferentes.
- A falta de padronização para sistema de escalonamento dificulta o reuso de código e adição de novos mecanismos de acesso.

Com base nas limitações encontradas e outras observações, são sugeridas propostas de melhorias e trabalhos futuros:

- É necessário continuar o processo de migração dos servidores de laboratório para a especificação e testá-los em plataforma compatível. Como foram seguidos os mesmos passos de desenvolvedores do projeto Go-Lab, infere-se que haverá compatibilidade para integração completa na plataforma desse projeto.
- Em relação aos problemas para a autorização de usuários é possível minimizá-lo utilizando mecanismo de autenticação *Single Sign-On* baseado em token e implementar fila com prioridade para usuários autenticados.
- O serviço de escalonamento pode ser extendido para dar suporte a reserva e a modelos de colaboração.
- Aplicações cliente baseadas em *widgets* compatíveis com a especificação podem ser exploradas para fornecer interfaces mais flexíveis, por exemplo, usando padrões como Gadgets Google e API OpenSocial. Além disso, seria ideal utilizar ferramentas de geração de código para interface gráfica.

- Mesmo com uma especificação comum, a área de desenvolvimento de laboratórios remotos carece de ferramentas mais próximas dos criadores para facilitar o desenvolvimento e manutenção de servidor de laboratório. Ferramentas como o Node-RED podem auxiliar na prototipação e desenvolvimento de aplicações mais robustas para uso em larga escala.

REFERÊNCIAS

AKTAN, B. et al. Distance learning applied to control engineering laboratories. **IEEE Transactions on education**, IEEE, v. 39, n. 3, p. 320–326, 1996.

ALVES, G. R. et al. Large and small scale networks of remote labs: a survey. **Advances on remote laboratories and e-learning experiences**, p. 15, 2007.

AUER, M. et al. Distributed virtual and remote labs in engineering. In: IEEE. **Industrial technology, 2003 IEEE international conference on**. [S.l.], 2003. v. 2, p. 1208–1213.

BOHUS, C. et al. **Running control engineering experiments over the Internet**. [S.l.], 1995.

BOSÁK, T.; ŽÁKOVÁ, K. Node. js based remote control of thermo-optical plant. In: IEEE. **Remote Engineering and Virtual Instrumentation (REV), 2015 12th International Conference on**. [S.l.], 2015. p. 209–213.

CAPUA, C. D.; LICCARDO, A.; MORELLO, R. On the web service-based remote didactical laboratory: further developments and improvements. In: IEEE. **2005 IEEE Instrumentationand Measurement Technology Conference Proceedings**. [S.l.], 2005. v. 3, p. 1692–1696.

CARDOZO, E. et al. A platform for networked robotics. In: IEEE. **Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on**. [S.l.], 2010. p. 1000–1005.

COLBRAN, S.; SCHULZ, M. An update to the software architecture of the ilab service broker. In: IEEE. **Remote Engineering and Virtual Instrumentation (REV), 2015 12th International Conference on**. [S.l.], 2015. p. 90–93.

COOPER, M.; FERREIRA, J. M. Remote laboratories extending access to science and engineering curricular. **Learning Technologies, IEEE Transactions on**, IEEE, v. 2, n. 4, p. 342–353, 2009.

CORTER, J. E. et al. Process and learning outcomes from remotely-operated, simulated, and hands-on student laboratories. **Computers & Education**, Elsevier, v. 57, n. 3, p. 2054–2067, 2011.

GARCÍA, M. L. et al. Rethinking remote laboratories: Widgets and smart devices. In: IEEE. **2013 IEEE Frontiers in Education Conference (FIE)**. [S.l.], 2013. p. 782–788.

GARCÍA-ZUBIA, J. et al. Addressing software impact in the design of remote laboratories. **Industrial Electronics, IEEE Transactions on**, IEEE, v. 56, n. 12, p. 4757–4767, 2009.

GOMES, L.; BOGOSYAN, S. Current trends in remote laboratories. **Industrial Electronics, IEEE Transactions on**, IEEE, v. 56, n. 12, p. 4744–4756, 2009.

GOVAERTS, S. **D4.1 Specifications of the Lab Owner and Cloud Services**. http://www.go-lab-project.eu/sites/default/files/files/deliverable/file/Go-Lab2014.

GRAVIER, C. et al. State of the art about remote laboratories paradigms-foundations of ongoing mutations. **International Journal of Online Engineering**, v. 4, n. 1, 2008.

GUSTAVSSON, I. et al. The visir project–an open source software initiative for distributed online laboratories. In: **REV 2007**. [S.l.: s.n.], 2007.

HALIMI, W.; GOVAERTS, S. **D4.5 Specifications of the Lab Owner and Cloud Services**. http://www.go-lab-project.eu/sites/default/files/files/deliverable/file/Go-Lab2015.

HALIMI, W.; SALZMANN, C.; GILLET, D. The mach-zehnder interferometer - a smart remote experiment based on a software template. In: IEEE. **2016 13th International Conference on Remote Engineering and Virtual Instrumentation (REV)**. [S.l.], 2016. p. 287–292.

HARWARD, V. J. et al. The ilab shared architecture: A web services infrastructure to build communities of internet accessible laboratories. **Proceedings of the IEEE**, IEEE, v. 96, n. 6, p. 931–950, 2008.

HERADIO, R. et al. Virtual and remote labs in education: A bibliometric analysis. **Computers & Education**, Elsevier, v. 98, p. 14–38, 2016.

IEEE. Working Draft Proposed Standard, **Standard for Networked Smart Learning Objects for Online Laboratories**. 2016. P1876?/D0.6 Draft.

JARA, C. A. et al. Hands-on experiences of undergraduate students in automatics and robotics using a virtual and remote laboratory. **Computers & Education**, Elsevier, v. 57, n. 4, p. 2451–2461, 2011.

JONG, T. de; SOTIRIOU, S.; GILLET, D. Innovations in stem education: the go-lab federation of online labs. **Smart Learning Environments**, Springer, v. 1, n. 1, p. 1–16, 2014.

LEI, K.; MA, Y.; TAN, Z. Performance comparison and evaluation of web development technologies in php, python, and node. js. In: IEEE. **Computational Science and Engineering (CSE), 2014 IEEE 17th International Conference on**. [S.l.], 2014. p. 661–668.

LIMA, J. P. C. et al. **RT2 - Avaliação dos resultados do protótipo do Grupo de Trabalho em Experimentação Remota Móvel**. [S.l.], sep 2015.

LOWE, D.; MACHET, T.; KOSTULSKI, T. Uts remote labs, labshare, and the sahara architecture. **Using Remote Labs in Education: Two Little Ducks in Remote Experimentation**, Universidad de Deusto, v. 8, p. 403, 2012.

LOWE, D. et al. Evolving remote laboratory architectures to leverage emerging internet technologies. **Learning Technologies, IEEE Transactions on**, IEEE, v. 2, n. 4, p. 289–294, 2009.

LOWE, D.; OROU, N. Interdependence of booking and queuing in remote laboratory scheduling. In: IEEE. **Remote engineering and virtual instrumentation (rev), 2012 9th international conference on**. [S.l.], 2012. p. 1–6.

LOWE, D. et al. Interoperating remote laboratory management systems (rlmss) for more efficient sharing of laboratory resources. **Computer Standards & Interfaces**, Elsevier, v. 43, p. 21–29, 2016.

MAITI, A.; KIST, A. A.; MAXWELL, A. D. Real-time remote access laboratory with distributed and modular design. **Industrial Electronics, IEEE Transactions on**, IEEE, v. 62, n. 6, p. 3607–3618, 2015.

MAITI, A.; MAXWELL, A. D.; KIST, A. A. An overview of system architectures for remote laboratories. In: IEEE. **Teaching, Assessment and Learning for Engineering (TALE), 2013 IEEE International Conference on**. [S.l.], 2013. p. 661–666.

NGOLO, M. et al. Architecture for remote laboratories based on rest web services. In: IEEE. **E-Learning in Industrial Electronics, 2009. ICELIE'09. 3rd IEEE International Conference on**. [S.l.], 2009. p. 30–35.

ORDUÑA, P. **Transitive and Scalable Federation Model for Remote Laboratory**. Tese (Doutorado) — Universidad de Deusto, 2013.

ORDUÑA, P. et al. Towards federated interoperable bridges for sharing educational remote laboratories. **Computers in Human Behavior**, Elsevier, v. 30, p. 389–395, 2014.

ORDUNA, P. et al. Generic integration of remote laboratories in public learning tools: organizational and technical challenges. In: IEEE. **2014 IEEE Frontiers in Education Conference (FIE) Proceedings**. [S.l.], 2014. p. 1–7.

ORDUÑA, P. et al. Adding new features to new and existing remote experiments through their integration in weblab-deusto. **iJOE**, v. 7, n. S2, p. 33–39, 2011.

ORDUNA, P. et al. Towards a microrlms approach for shared development of remote laboratories. In: IEEE. **Remote Engineering and Virtual Instrumentation (REV), 2014 11th International Conference on**. [S.l.], 2014. p. 375–381.

PAYNE, L. J.; SCHULZ, M. F. Java implementation of the batched ilab shared architecture. In: IEEE. **Remote Engineering and Virtual Instrumentation (REV), 2013 10th International Conference on**. [S.l.], 2013. p. 1–3.

RICHTER, T.; GRUBE, P.; ZUTIN, D. A standardized metadata set for annotation of virtual and remote laboratories. In: IEEE. **Multimedia (ISM), 2012 IEEE International Symposium on**. [S.l.], 2012. p. 451–456.

RICHTER, T.; TETOUR, Y.; BOEHRINGER, D. Library of labs-a european project on the dissemination of remote experiments and virtual laboratories. In: **ISM**. [S.l.: s.n.], 2011. p. 543–548.

RUIZ, E. S. C. et al. Widget and smart devices. a different aproach for online learning scenarios. In: **Proceedings of the 4th IEEE Global Engineering Education Conference (EDUCON)**. [S.l.: s.n.], 2013.

SALIAH-HASSANE, H. et al. A general framework for web services and grid-based technologies for online laboratories. 2005.

SALZMANN, C.; GILLET, D. Smart device paradigm standardization for online labs. In: **4th IEEE Global Engineering Education Conference (EDUCON)**. [S.l.: s.n.], 2013.

SANCRISTOBAL, E. et al. Integration view of web labs and learning management systems. In: IEEE. **IEEE EDUCON 2010 Conference**. [S.l.], 2010. p. 1409–1417.

SANCRISTOBAL, E. et al. Widget and smart devices. A different aproach for online learning scenarios BT - 2013 IEEE Global Engineering Education Conference, EDUCON 2013, March 13, 2013 - March 15, 2013. p. 808–812, 2013. Disponível em: <http://dx.doi.org/10.1109/EduCon.2013.6530199>.

SANTANA, I. et al. Remote laboratories for education and research purposes in automatic control systems. **Industrial Informatics, IEEE Transactions on**, IEEE, v. 9, n. 1, p. 547–556, 2013.

SIMAO, J. P. S. **RELLE: Sistema de Gerenciamento de Experimentos Remotos**. fev 2016. Trabalho de Conclusão de Curso (Graduação em Tecnologias da Informação e Comunicação).

SOMMERVILLE, I. Software engineering. **ed: Addison Wesley**, 2004.

TAWFIK, M. et al. Laboratory as a Service (LaaS): a Model for Developing and Implementing Remote Laboratories as Modular Components. n. February, p. 11–20, 2014.

TAWFIK, M. et al. Virtual instrument systems in reality (visir) for remote wiring and measurement of electronic circuits on breadboard. **IEEE Transactions on Learning Technologies**, IEEE, v. 6, n. 1, p. 60–72, 2013.

THOMPSON, C. W. Smart devices and soft controllers. **IEEE Internet Computing**, IEEE, v. 9, n. 1, p. 82–85, 2005.

TILKOV, S.; VINOSKI, S. Node. js: Using javascript to build high-performance network programs. **IEEE Internet Computing**, IEEE Computer Society, v. 14, n. 6, p. 80, 2010.

TRÖGER, P. et al. Soa meets robots-a service-based software infrastructure for remote laboratories. **iJOE**, v. 4, n. 2, p. 24–30, 2008.

WANG, N. et al. Developing a remote laboratory at tamuq based on a novel unified framework. **age**, v. 26, p. 1, 2015.

YEUNG, H.; LOWE, D.; MURRAY, S. Interoperability of remote laboratories systems. **International Journal of Online Engineering (iJOE)**, v. 6, n. 5, p. 71–80, 2010.

ZUTIN, D. G. et al. Lab2go?a repository to locate educational online laboratories. In: IEEE. **IEEE EDUCON 2010 Conference**. [S.l.], 2010. p. 1741–1746.